EcoFuzz: Adaptive Energy-Saving Greybox Fuzzing as a Variant of the Adversarial Multi-Armed Bandit

Tai Yue Pengfei Wang Yong Tang Enze Wang

Bo Yu Kai Lu Xu Zhou

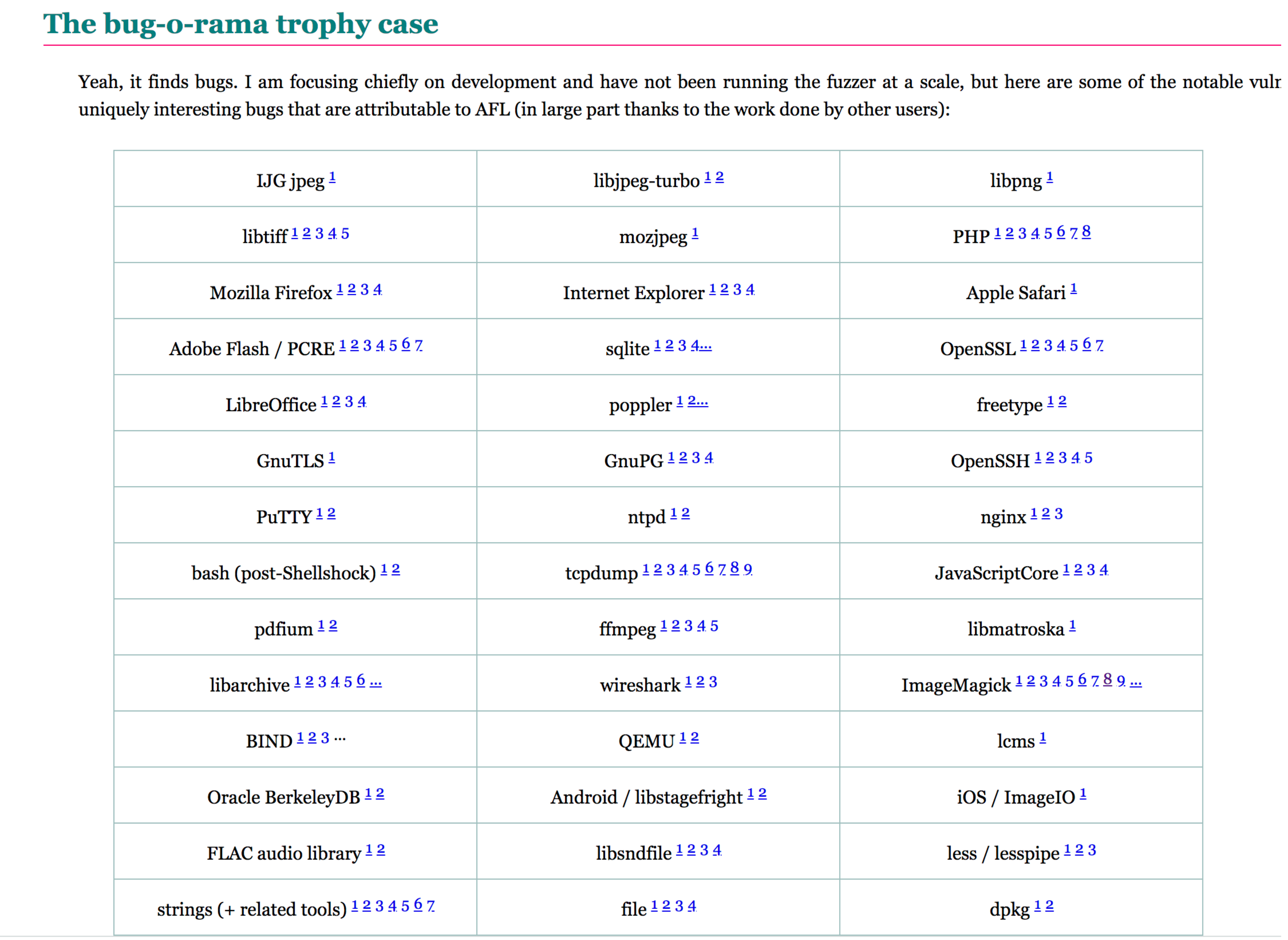
Email: yuetai17@nudt.edu.cn

EcoFuzz: https://github.com/MoonLight-SteinsGate/EcoFuzz

National University of Defense Technology

Coverage-based Greybox Fuzzing

- Effective approach for identifying vulnerabilities
- American Fuzzy Lop (AFL)



The bug-o-rama trophy case

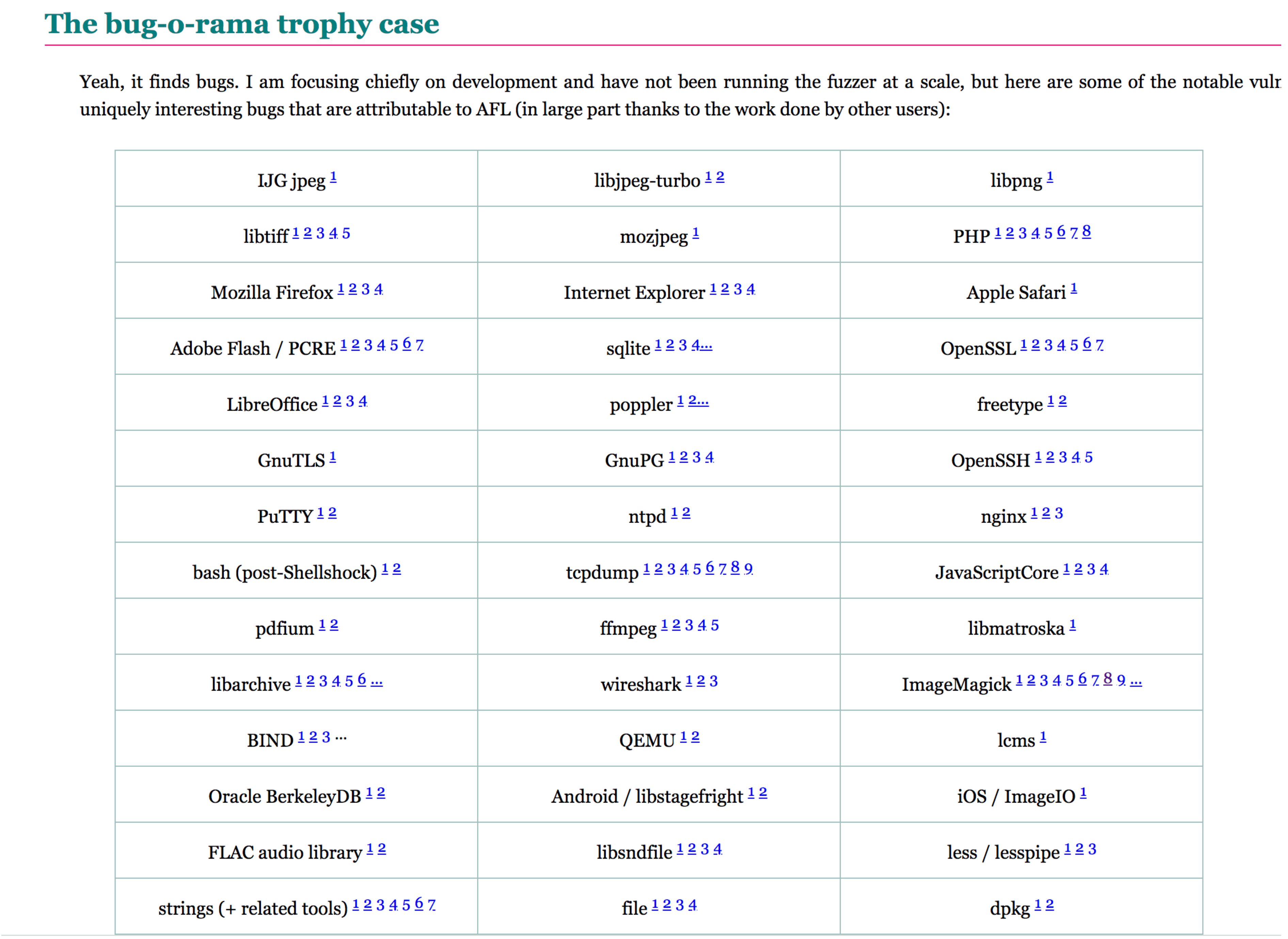
Yeah, it finds bugs. I am focusing chiefly on development and have not been running the fuzzer at a scale, but here are some of the notable vulr uniquely interesting bugs that are attributable to AFL (in large part thanks to the work done by other users):

IJG jpeg 1	libjpeg-turbo 1 2	libpng 1
libtiff 1 2 3 4 5	mozjpeg 1	PHP 1 2 3 4 5 6 7 8
Mozilla Firefox 1 2 3 4	Internet Explorer 1 2 3 4	Apple Safari 1
Adobe Flash / PCRE 1 2 3 4 5 6 7	sqlite 1 2 3 4...	OpenSSL 1 2 3 4 5 6 7
LibreOffice 1 2 3 4	poppler 1 2...	freetype 1 2
GnuTLS 1	GnuPG 1 2 3 4	OpenSSH 1 2 3 4 5
PuTTY 1 2	ntpd 1 2	nginx 1 2 3
bash (post-Shellshock) 1 2	tcpdump 1 2 3 4 5 6 7 8 9	JavaScriptCore 1 2 3 4
pdfium 1 2	ffmpeg 1 2 3 4 5	libmatroska 1
libarchive 1 2 3 4 5 6 ...	wireshark 1 2 3	ImageMagick 1 2 3 4 5 6 7 8 9 ...
BIND 1 2 3 ...	QEMU 1 2	lcms 1
Oracle BerkeleyDB 1 2	Android / libstagefright 1 2	iOS / ImageIO 1
FLAC audio library 1 2	libsndfile 1 2 3 4	less / lesspipe 1 2 3
strings (+ related tools) 1 2 3 4 5 6 7	file 1 2 3 4	dpkg 1 2

The bugs found by AFL

Coverage-based Greybox Fuzzing

- Effective approach for identifying vulnerabilities
- American Fuzzy Lop (AFL)
  - Mutation operator (MOPT, FairFuzz)
  - Initial seeds (Skyfire)



The bug-o-rama trophy case

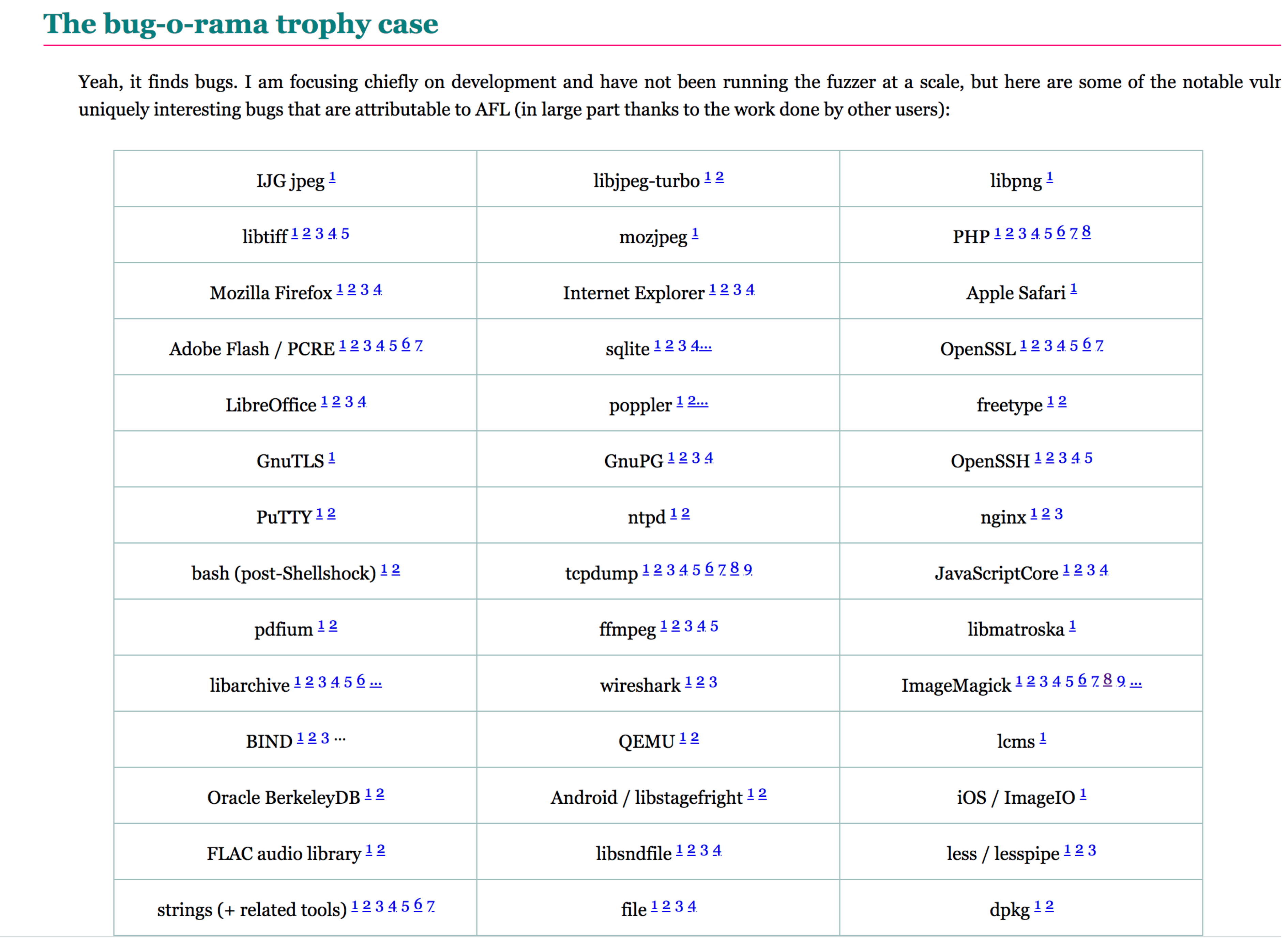
Yeah, it finds bugs. I am focusing chiefly on development and have not been running the fuzzer at a scale, but here are some of the notable vulr uniquely interesting bugs that are attributable to AFL (in large part thanks to the work done by other users):

IJG jpeg 1	libjpeg-turbo 1 2	libpng 1
libtiff 1 2 3 4 5	mozjpeg 1	PHP 1 2 3 4 5 6 7 8
Mozilla Firefox 1 2 3 4	Internet Explorer 1 2 3 4	Apple Safari 1
Adobe Flash / PCRE 1 2 3 4 5 6 7	sqlite 1 2 3 4...	OpenSSL 1 2 3 4 5 6 7
LibreOffice 1 2 3 4	poppler 1 2...	freetype 1 2
GnuTLS 1	GnuPG 1 2 3 4	OpenSSH 1 2 3 4 5
PuTTY 1 2	ntpd 1 2	nginx 1 2 3
bash (post-Shellshock) 1 2	tcpdump 1 2 3 4 5 6 7 8 9	JavaScriptCore 1 2 3 4
pdfium 1 2	ffmpeg 1 2 3 4 5	libmatroska 1
libarchive 1 2 3 4 5 6 ...	wireshark 1 2 3	ImageMagick 1 2 3 4 5 6 7 8 9 ...
BIND 1 2 3 ...	QEMU 1 2	lcms 1
Oracle BerkeleyDB 1 2	Android / libstagefright 1 2	iOS / ImageIO 1
FLAC audio library 1 2	libsndfile 1 2 3 4	less / lesspipe 1 2 3
strings (+ related tools) 1 2 3 4 5 6 7	file 1 2 3 4	dpkg 1 2

The bugs found by AFL

Coverage-based Greybox Fuzzing

- Effective approach for identifying vulnerabilities
- American Fuzzy Lop (AFL)
  - Mutation operator (MOPT, FairFuzz)
  - Initial seeds (Skyfire)



The bug-o-rama trophy case

Yeah, it finds bugs. I am focusing chiefly on development and have not been running the fuzzer at a scale, but here are some of the notable vulr uniquely interesting bugs that are attributable to AFL (in large part thanks to the work done by other users):

IJG jpeg 1	libjpeg-turbo 1 2	libpng 1
libtiff 1 2 3 4 5	mozjpeg 1	PHP 1 2 3 4 5 6 7 8
Mozilla Firefox 1 2 3 4	Internet Explorer 1 2 3 4	Apple Safari 1
Adobe Flash / PCRE 1 2 3 4 5 6 7	sqlite 1 2 3 4...	OpenSSL 1 2 3 4 5 6 7
LibreOffice 1 2 3 4	poppler 1 2...	freetype 1 2
GnuTLS 1	GnuPG 1 2 3 4	OpenSSH 1 2 3 4 5
PuTTY 1 2	ntpd 1 2	nginx 1 2 3
bash (post-Shellshock) 1 2	tcpdump 1 2 3 4 5 6 7 8 9	JavaScriptCore 1 2 3 4
pdfium 1 2	ffmpeg 1 2 3 4 5	libmatroska 1
libarchive 1 2 3 4 5 6 ...	wireshark 1 2 3	ImageMagick 1 2 3 4 5 6 7 8 9 ...
BIND 1 2 3 ...	QEMU 1 2	lcms 1
Oracle BerkeleyDB 1 2	Android / libstagefright 1 2	iOS / ImageIO 1
FLAC audio library 1 2	libsndfile 1 2 3 4	less / lesspipe 1 2 3
strings (+ related tools) 1 2 3 4 5 6 7	file 1 2 3 4	dpkg 1 2

The bugs found by AFL

Motivation

- Shortcomings in schedule algorithm
  - Assign too much **energy** on seeds exercising high-frequency paths
  - Simple select strategy
- Few works focus on this
  - AFLFast
- Limitation of current model

Motivation

- Shortcomings in schedule algorithm
  - Assign too much energy on seeds exercising high-frequency paths
  - Simple select strategy
- Few works focus on this
  - AFLFast
- Limitation of current model

Motivation

- Shortcomings in schedule algorithm
  - Assign too much energy on seeds exercising high-frequency paths
  - Simple select strategy
- Few works focus on this
  - AFLFast
- Limitation of current model

Motivation

- Proposing a new model
- Improving schedule algorithm
  - Search strategy: selecting which seed
  - Power schedule: assigning how many energy

Motivation

- Proposing a new model
- Improving schedule algorithm
    - Search strategy: selecting which seed
    - Power schedule: assigning how many energy

Contributions

- One model: a variant of the Adversarial Multi-Armed Bandit (VAMAB)

- One tool: an adaptive energy-saving fuzzer named EcoFuzz

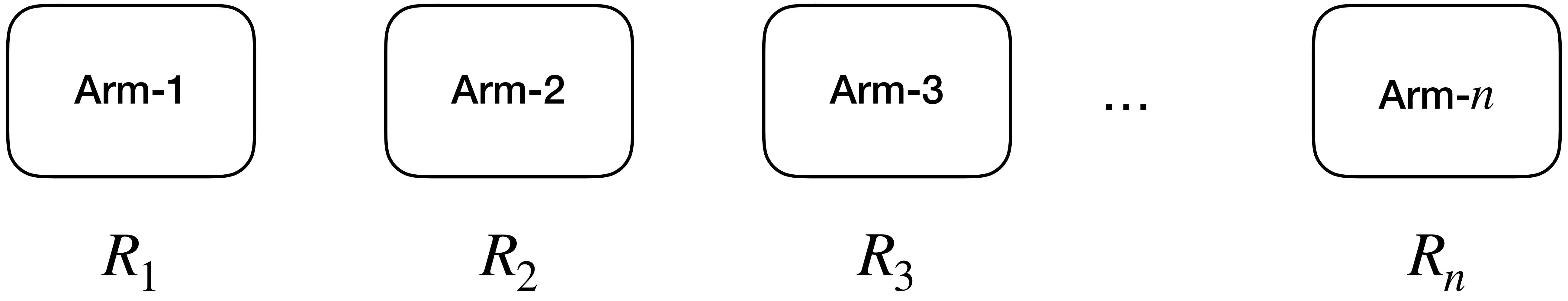
- Comprehensive evaluation: a serial of experiments from different metrics

Contributions

- One model: a variant of the Adversarial Multi-Armed Bandit (VAMAB)
- One tool: an adaptive energy-saving fuzzer named EcoFuzz
- Comprehensive evaluation: a serial of experiments from different metrics

Classical Multi-Armed Bandit

- Constant number of arms
- Reward
- **Reward probability**
  - constant and unknown
- Target
  - maximizing the rewards in finite trials



Arm-1 Arm-2 Arm-3 … Arm-n

R_1 R_2 R_3 R_n

Classical Multi-Armed Bandit

Classical MAB	CGF
• Arms	• Seeds
• Reward	• Finding a new path
• Maximize the rewards	• Maximize path coverage

Classical Multi-Armed Bandit

Classical MAB

- Arms
- Reward
- Maximize the rewards
- The number of arms is constant
- The reward probability is constant

CGF

- Seeds
- Finding a new path
- Maximize path coverage
- The number of seeds is variable
- The probability of finding new paths is decreasing

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

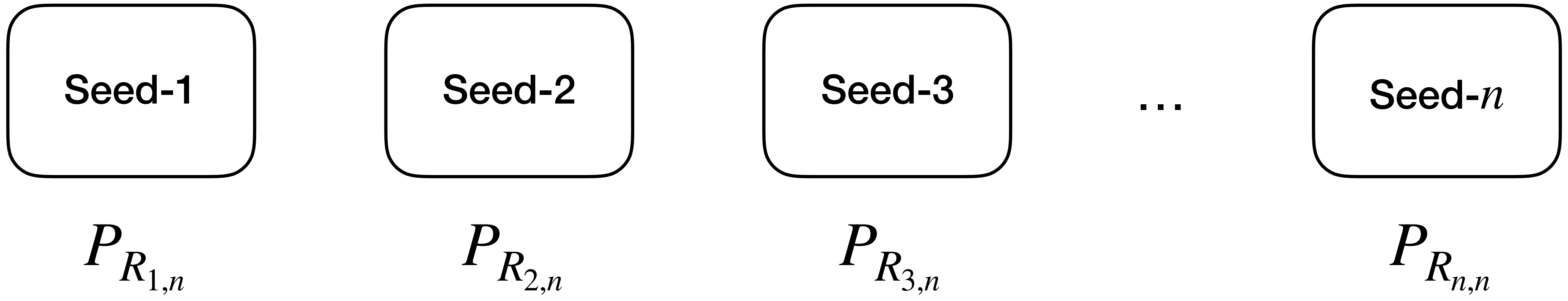
VAMAB

- Arms (seeds)
- Reward (Finding a new path)
- Maximize the rewards (path coverage) in finite trails
- The number of arms is variable (increasing), with a upper bound of n_p
- The **reward probability, which is the probability to find new paths,** is variable (decreasing)

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

VAMAB

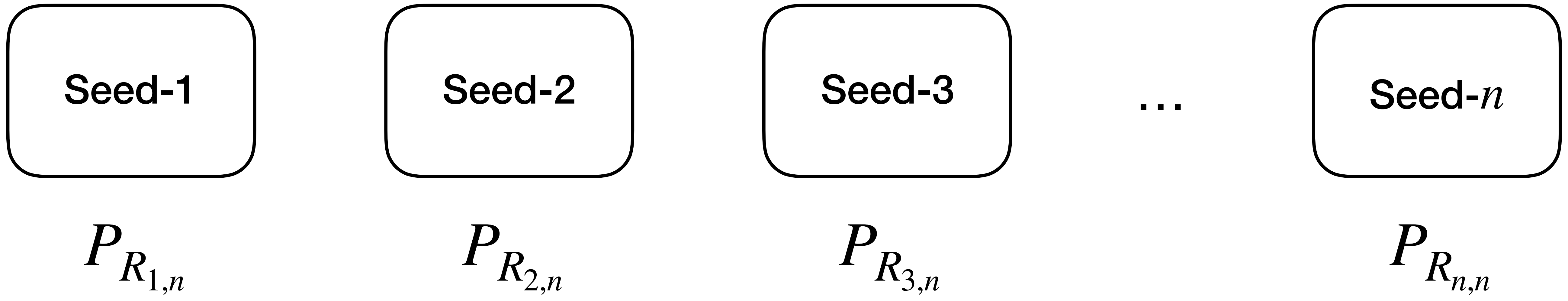
- Arms (seeds)
- Reward (Finding a new path)
- Maximize the rewards (path coverage) in finite trails
- The number of arms is variable (increasing), with a upper bound of n_p
- The **reward probability, which is the probability to find new paths,** is variable (decreasing)



A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

VAMAB

- Arms (seeds)
- Reward (Finding a new path)
- Maximize the rewards (path coverage) in finite trails
- The number of arms is variable (increasing), with a upper bound of n_p
- The **reward probability, which is the probability to find new paths,** is variable (decreasing)

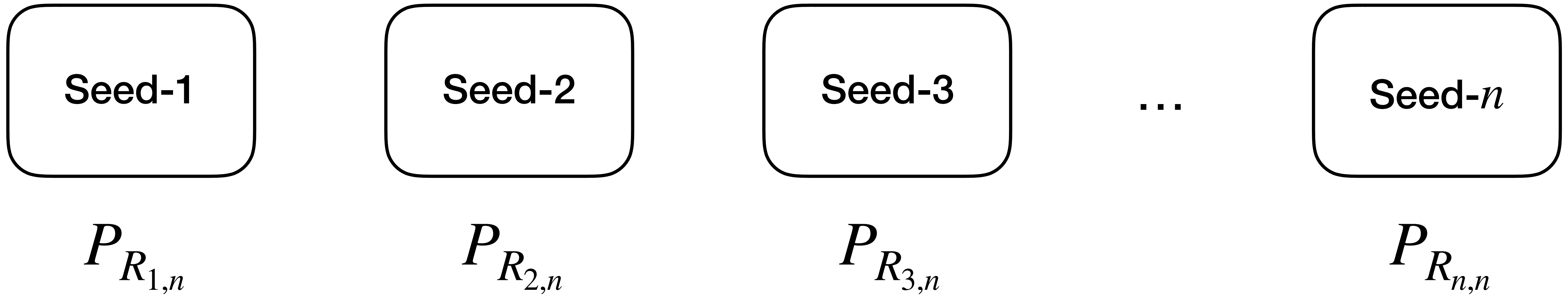


$$P_{R_{i,n}} = \sum_{j=n+1}^{n_p} p_{ij} = 1 - \sum_{j=1}^{n} p_{ij}$$

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

VAMAB

- Arms (seeds)
- Reward (Finding a new path)
- Maximize the rewards (path coverage) in finite trails
- The number of arms is variable (increasing), with a upper bound of n_p
- The **reward probability, which is the probability to find new paths,** is variable (decreasing)



Seed-1 Seed-2 Seed-3 … Seed-n

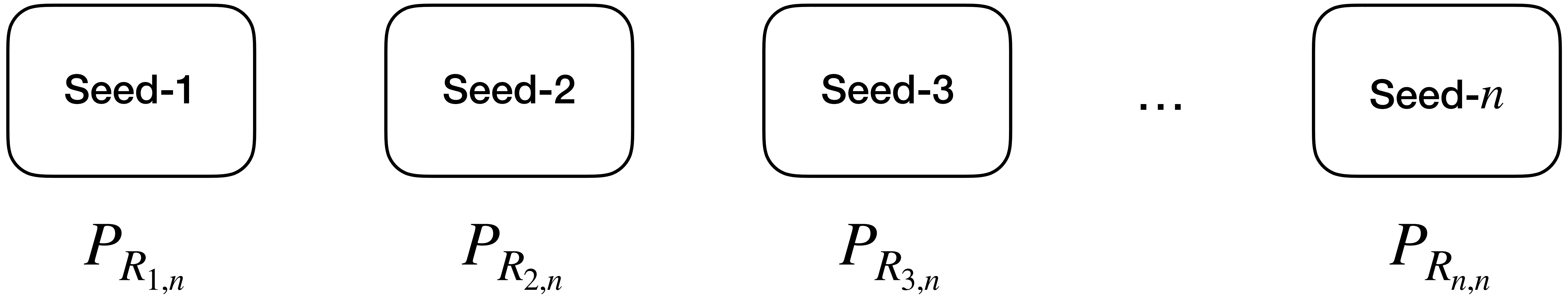
$P_{R_{1,n}}$ $P_{R_{2,n}}$ $P_{R_{3,n}}$ $P_{R_{n,n}}$

$$P_{R_{i,n}} = \sum_{j=n+1}^{\overset{\textbf{Total Paths}}{\overset{\downarrow}{n_p}}} p_{ij} = 1 - \sum_{j=1}^{n} p_{ij}$$

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

VAMAB

- Arms (seeds)
- Reward (Finding a new path)
- Maximize the rewards (path coverage) in finite trails
- The number of arms is variable (increasing), with a upper bound of n_p
- The **reward probability, which is the probability to find new paths,** is variable (decreasing)



Total Paths

$$P_{R_{i,n}} = \sum_{j=n+1}^{n_p} p_{ij} = 1 - \sum_{j=1}^{n} p_{ij}$$

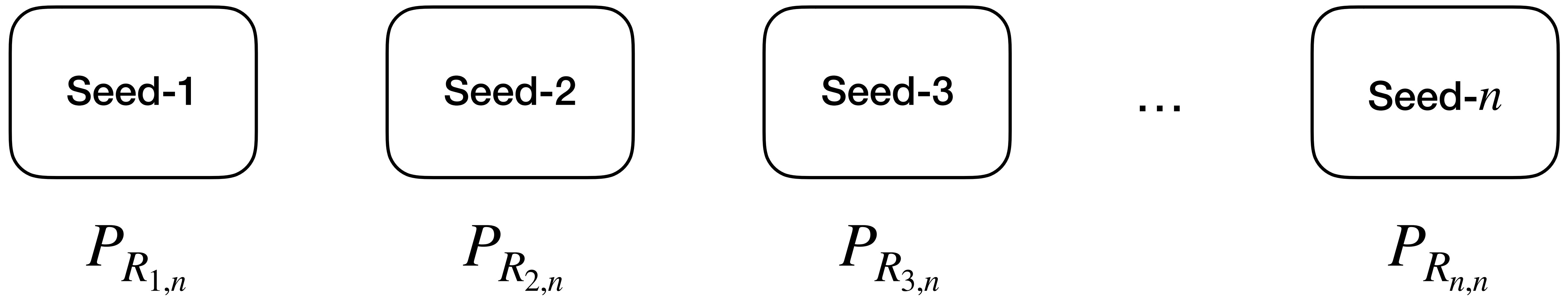
Transition Probability

- The probability of mutating the seed i to generate a test case executing the path j.

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

VAMAB

- Arms (seeds)
- Reward (Finding a new path)
- Maximize the rewards (path coverage) in finite trails
- The number of arms is variable (increasing), with a upper bound of n_p
- The **reward probability, which is the probability to find new paths,** is variable (decreasing)



Seed-1 Seed-2 Seed-3 … Seed-n

$P_{R_{1,n}}$ $P_{R_{2,n}}$ $P_{R_{3,n}}$ $P_{R_{n,n}}$

Total Paths

$$P_{R_{i,n}} = \sum_{j=n+1}^{n_p} p_{ij} = 1 - \sum_{j=1}^{n} p_{ij} \Longrightarrow P_{R_{i,n}} > P_{R_{i,n+1}} > \ldots > P_{R_{i,n_p}} = 0$$

Transition Probability

- The probability of mutating the seed i to generate a test case executing the path j.

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

VAMAB

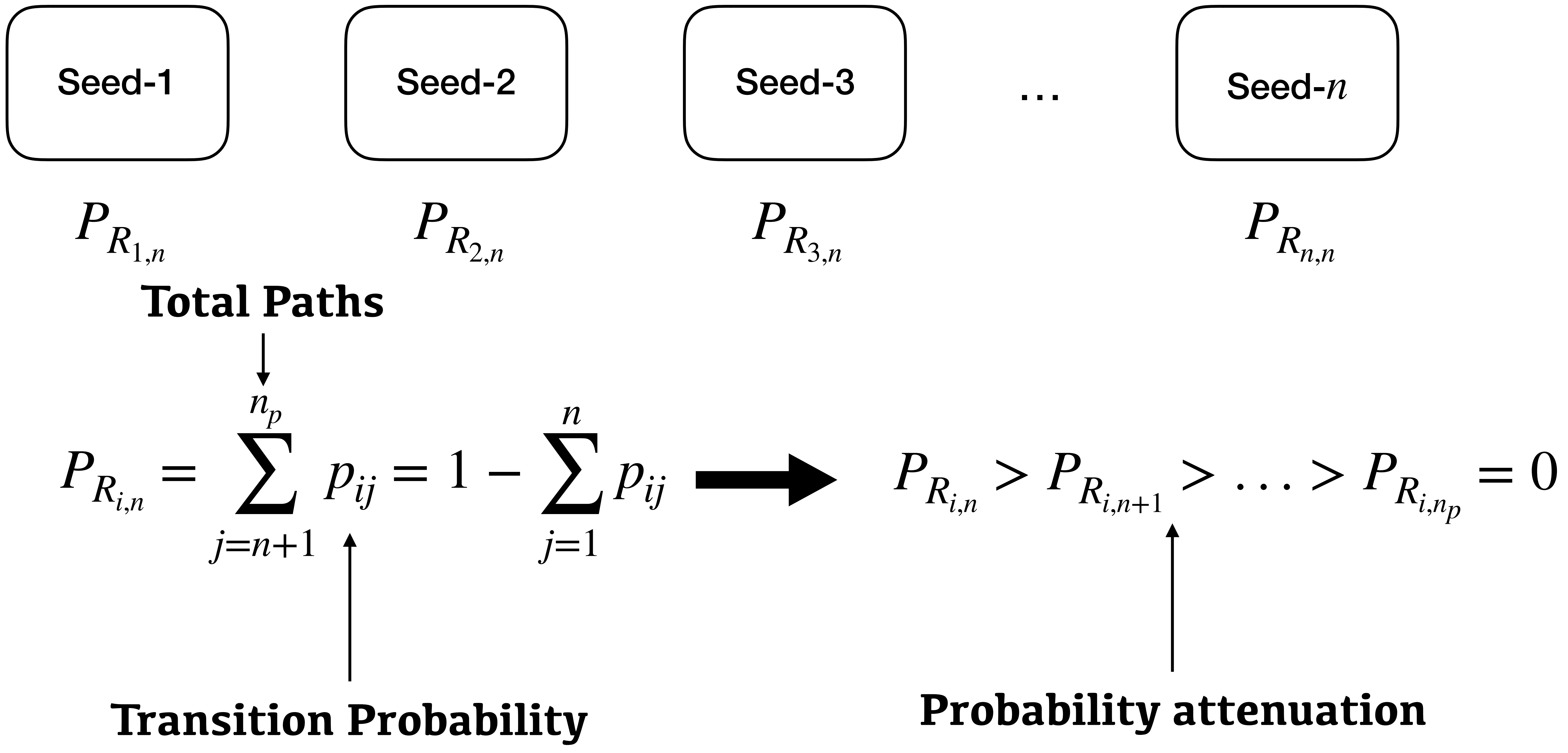
- Arms (seeds)

- Reward (Finding a new path)

- Maximize the rewards (path coverage) in finite trails

- The number of arms is variable (increasing), with a upper bound of n_p

- The **reward probability, which is the probability to find new paths,** is variable (decreasing)



$$P_{R_{i,n}} = \sum_{j=n+1}^{n_p} p_{ij} = 1 - \sum_{j=1}^{n} p_{ij} \longrightarrow P_{R_{i,n}} > P_{R_{i,n+1}} > \ldots > P_{R_{i,n_p}} = 0$$

- The probability of mutating the seed i to generate a test case executing the path j.

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

Exploration

- Estimate their reward probabilities

Exploitation

- Select the seed with a high reward probability

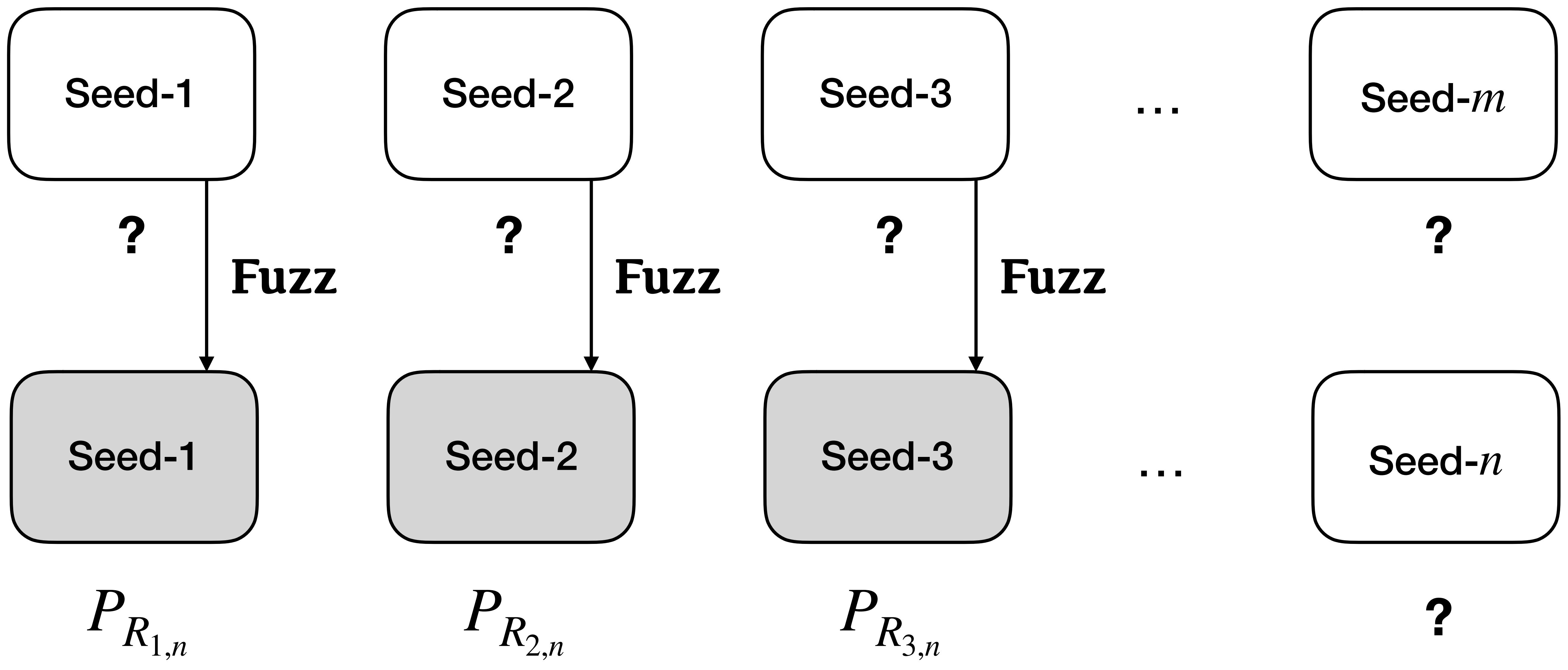
A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

Exploration

- Estimate their **reward probabilities**

Exploitation

- Select the seed with a high reward probability



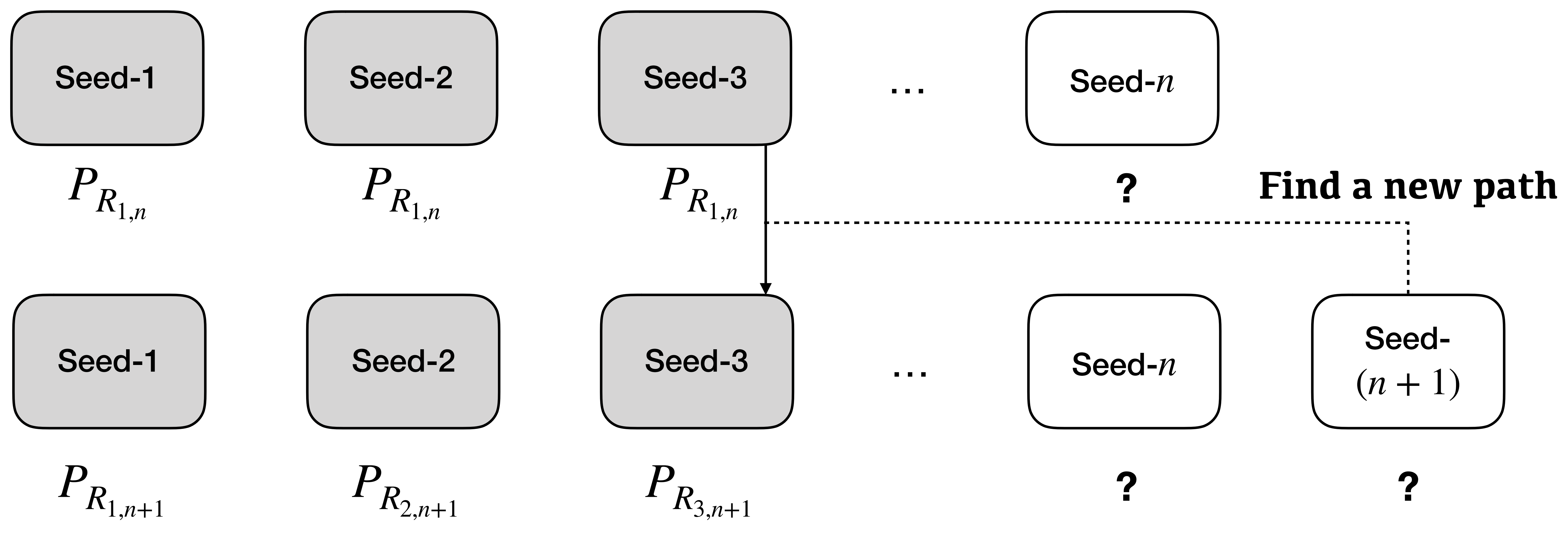
A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

Exploration

- Estimate their reward probabilities

Exploitation

- Select the seed with a high **reward probability**



Seed-1 Seed-2 Seed-3 … Seed-n

$P_{R_{1,n}}$ $P_{R_{1,n}}$ $P_{R_{1,n}}$?

Find a new path

Seed-1 Seed-2 Seed-3 … Seed-n Seed-$(n+1)$

$P_{R_{1,n+1}}$ $P_{R_{2,n+1}}$ $P_{R_{3,n+1}}$? ?

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

Exploration

- Estimate their reward probabilities
- Focusing on exploring new seeds:
  - Assigning fewer energy on the old seeds with high **reward probabilities**

Exploitation

- Select the seed with a high reward probability
- Focusing on exploiting old seeds:
  - Missing some new seeds with higher reward probabilities

A Variant of the Adversarial Multi-Armed Bandit (VAMAB)

Exploration

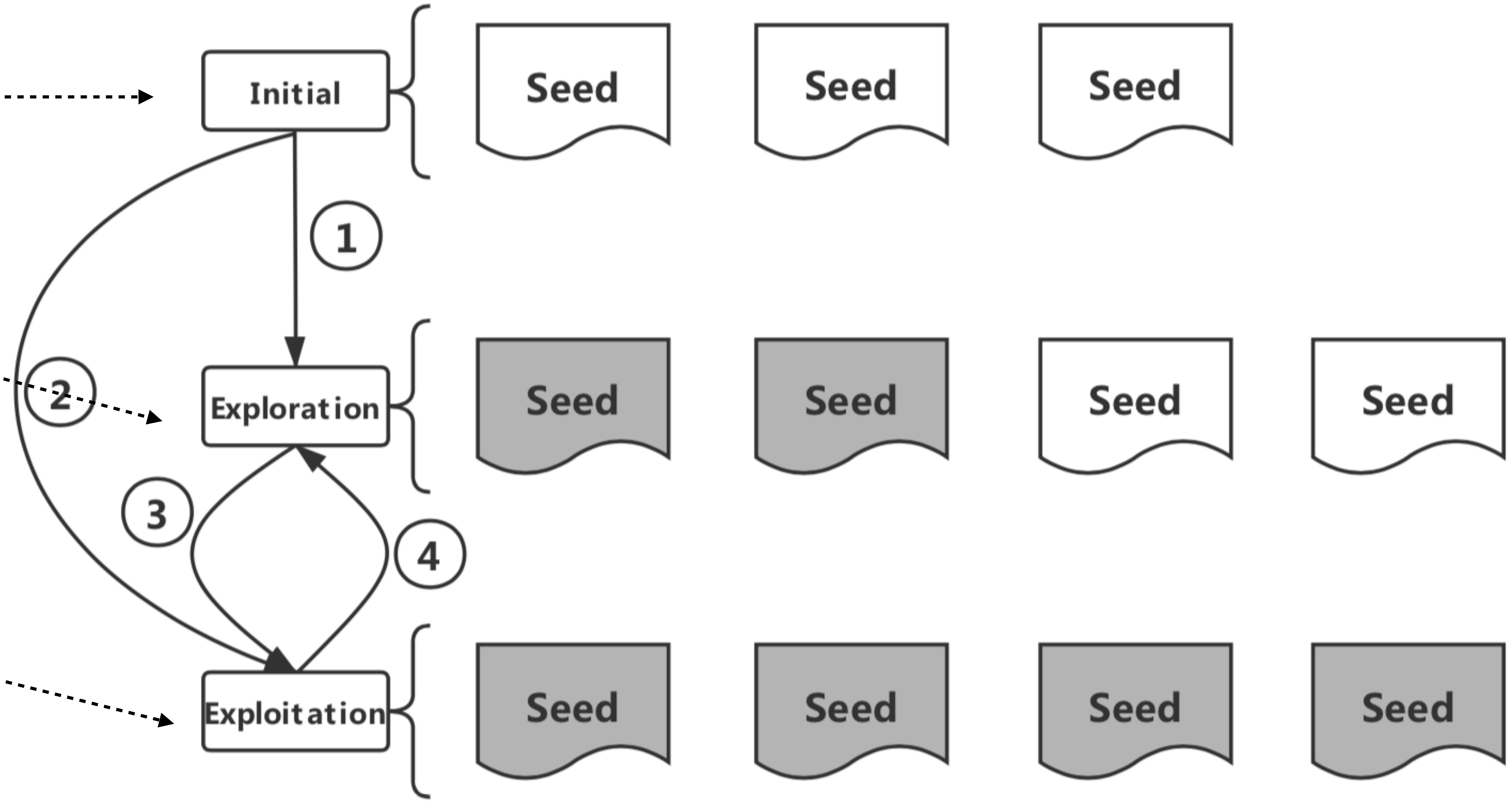
- Estimate their reward probabilities
- Focusing on exploring new seeds:
  - Assigning fewer energy on the old seeds with high reward probabilities

Exploitation

- Select the seed with a high reward probability
- Focusing on exploiting old seeds:
  - Missing some new seeds with higher **reward probabilities**

Three States in CGF

- **Initial state**: all seeds are unfuzzed
- **Exploration state**: part of seeds in the seed queue are fuzzed
- **Exploitation State**: all seeds in the seed queue have been fuzzed



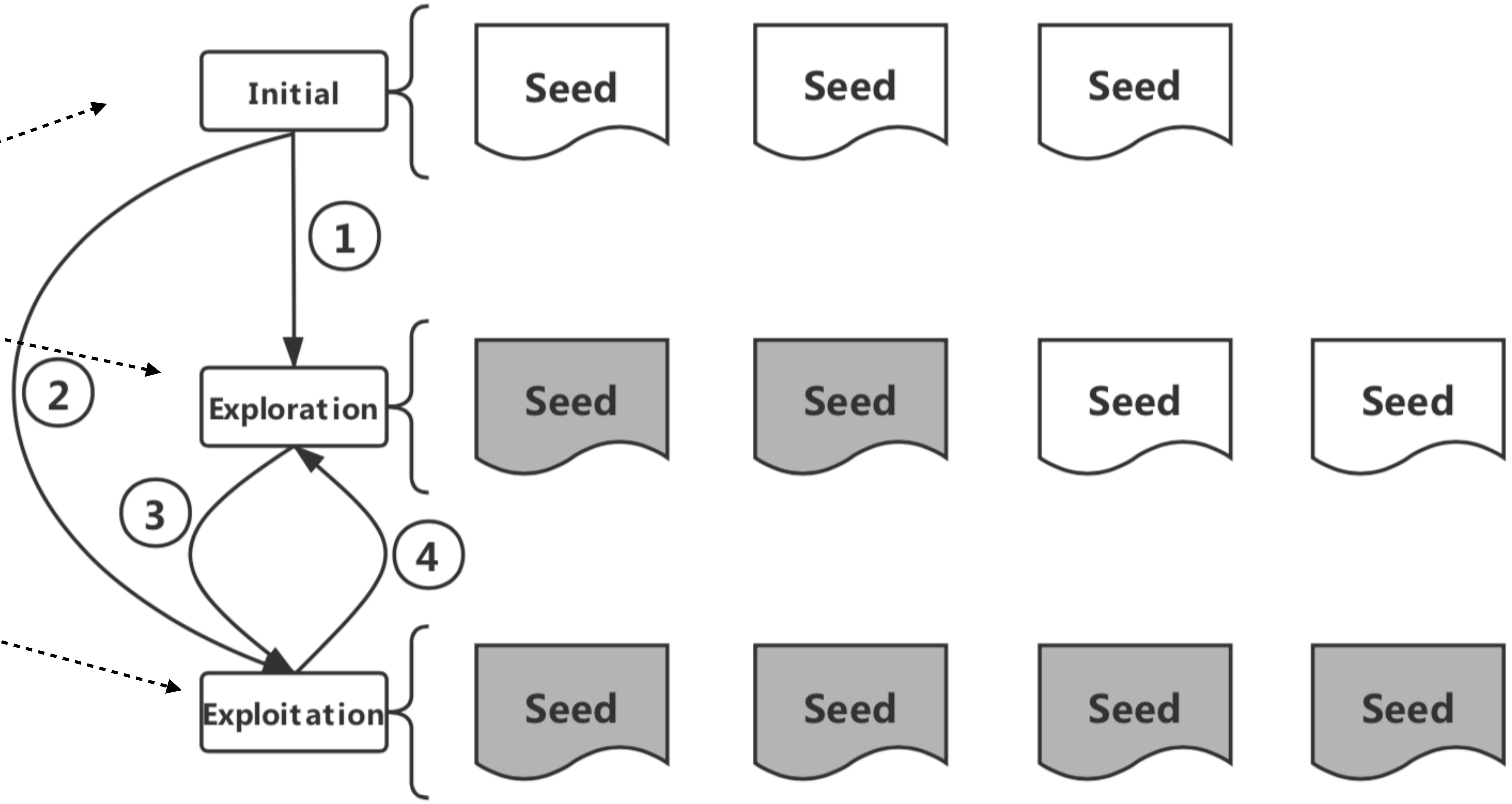
How to Maximize Coverage

Search Strategy

- Estimating the **reward probability**
- Selecting the seeds with high **reward probabilities**

Power Schedule

- Avoiding assigning too much energy to some seeds



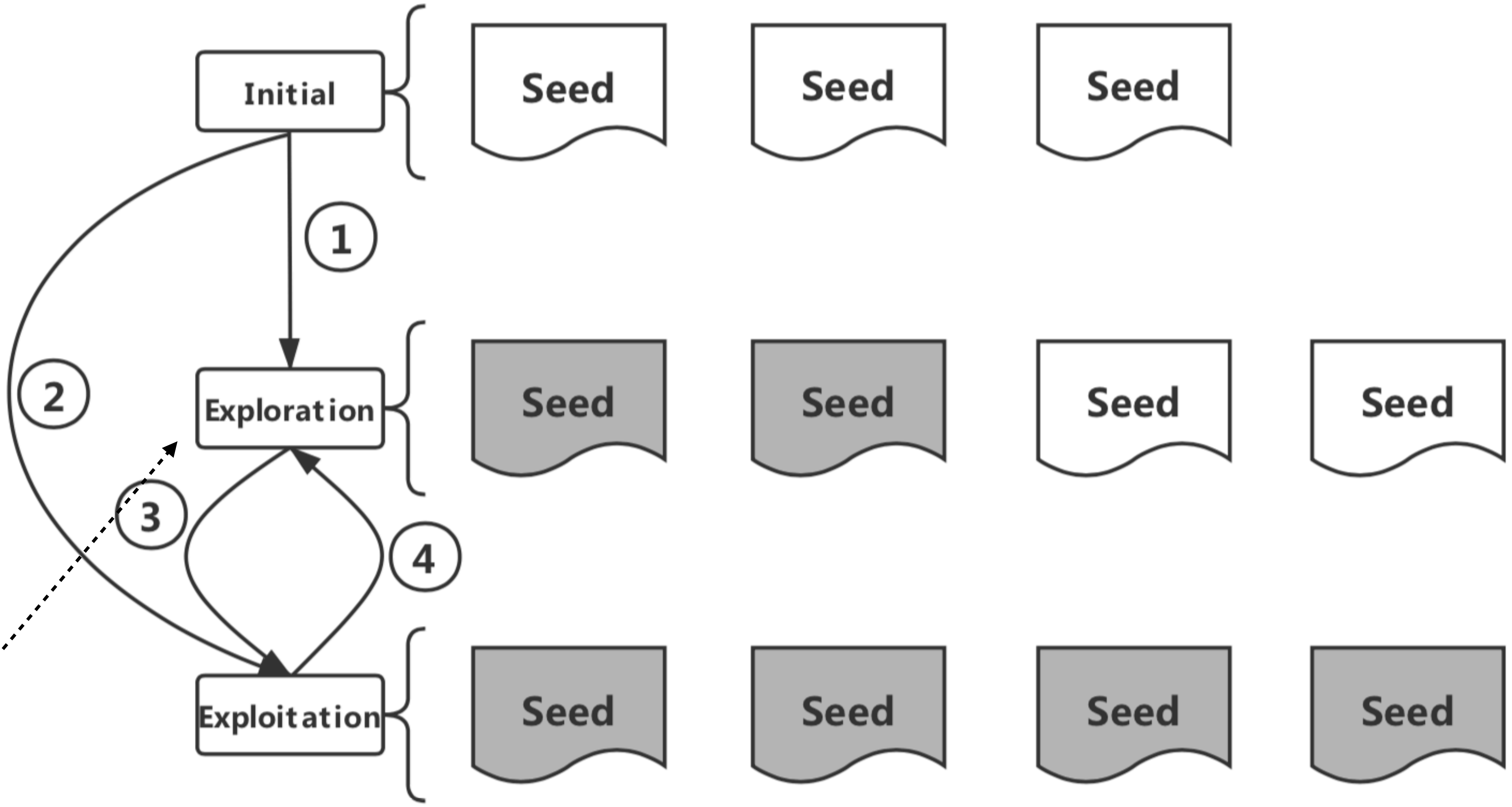
How to Maximize Coverage

Search Strategy

- Estimating the reward probability
- Selecting the seeds with high reward probabilities

Power Schedule

- Avoiding assigning too much energy on some seeds



Contributions

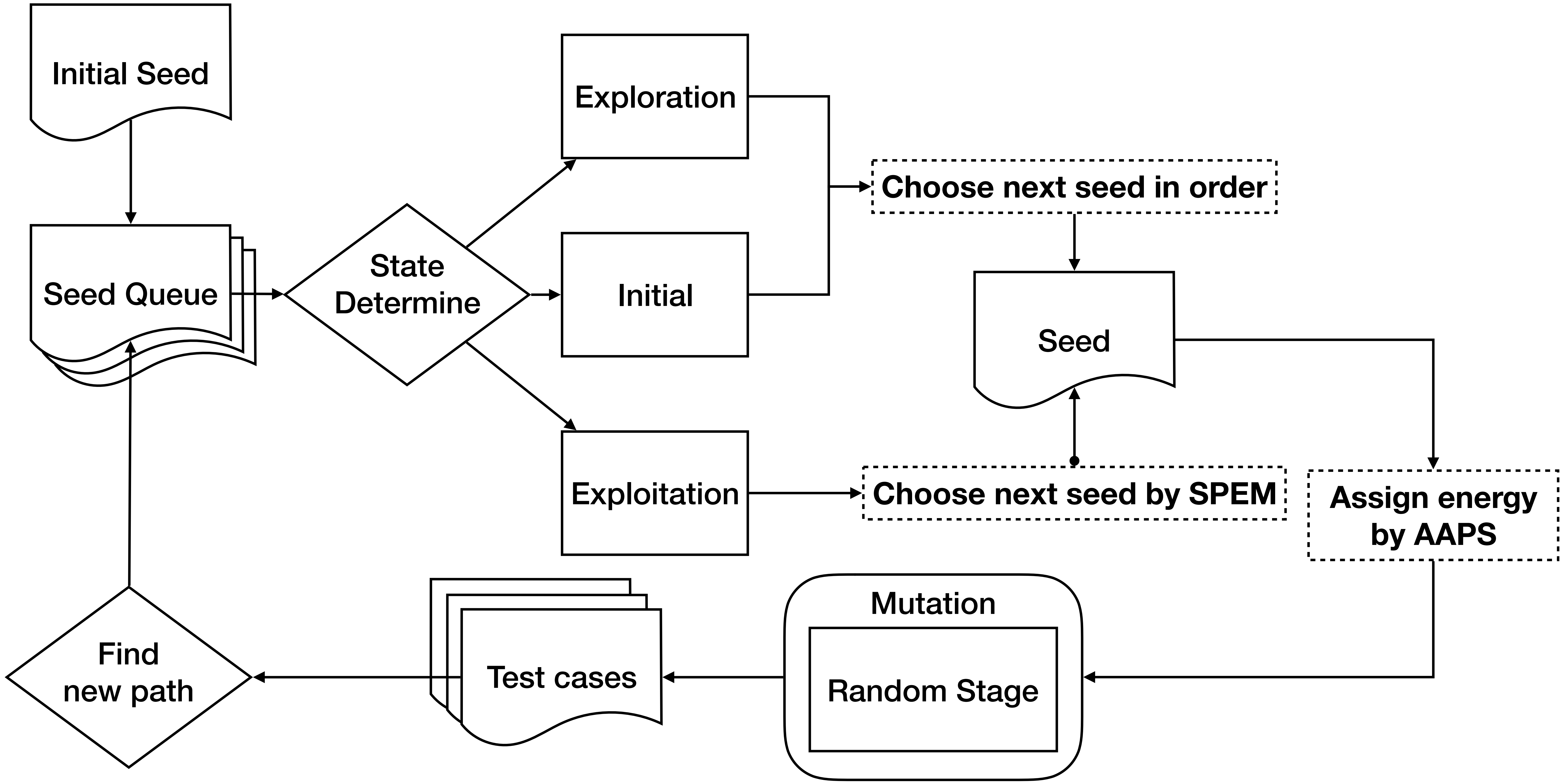
- One model: a variant of the Adversarial Multi-Armed Bandit (VAMAB)

- One tool: an adaptive energy-saving fuzzer named EcoFuzz

- Comprehensive evaluation: a serial of experiments from different metrics

EcoFuzz

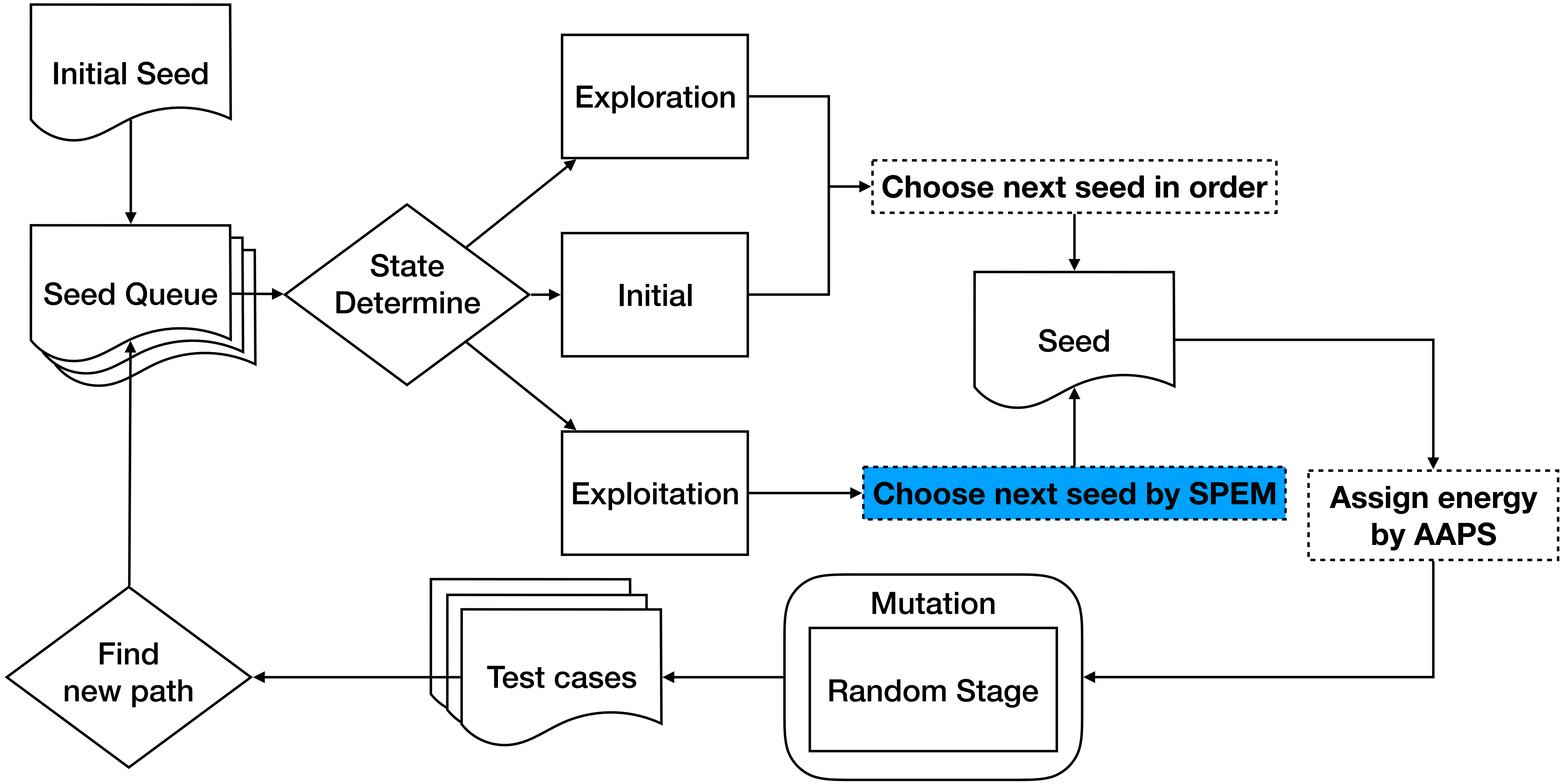
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

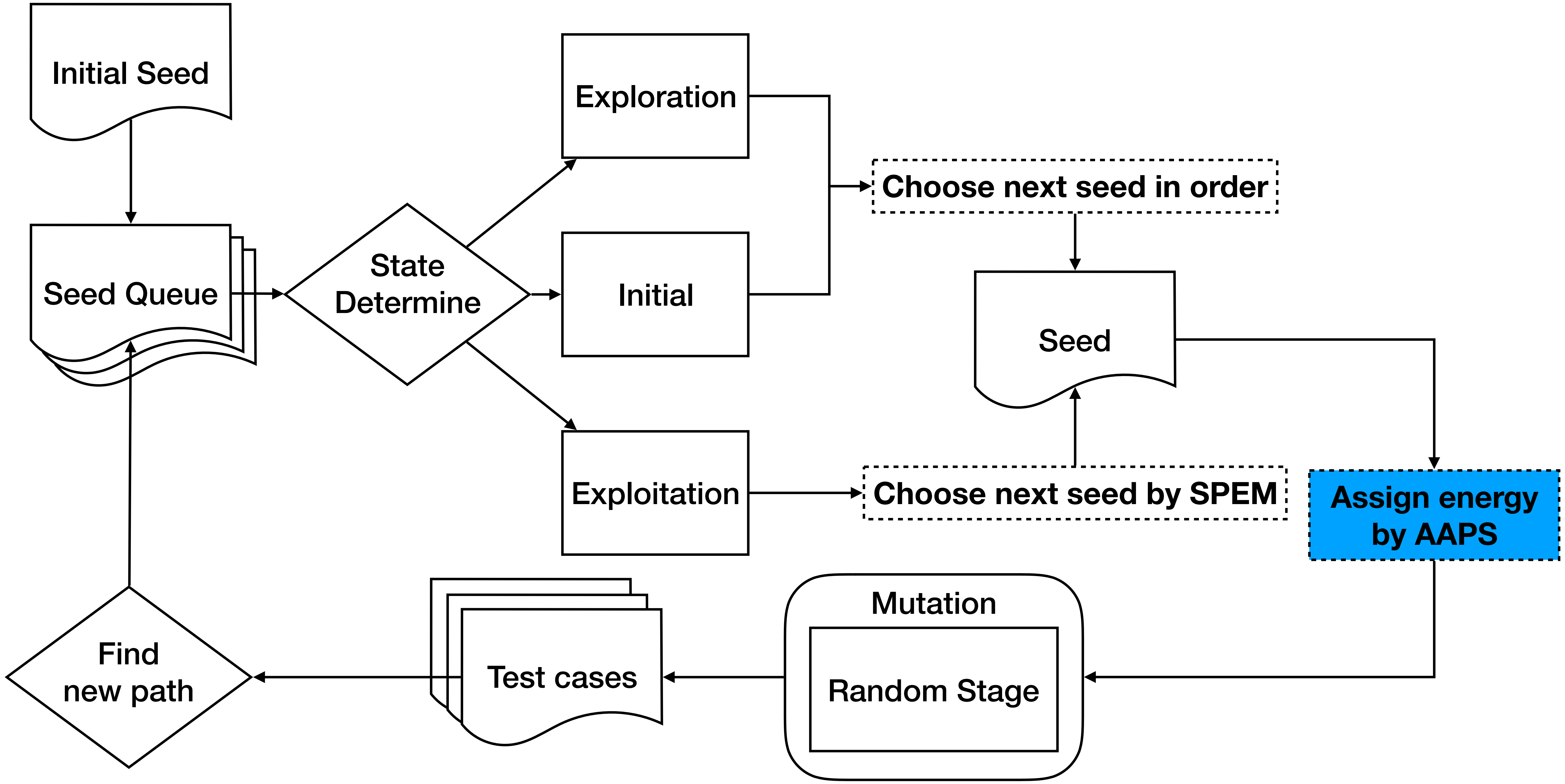
- Main Framework



- Based on AFL
- Search strategy: **Self-transition-based Probability Estimation Method (SPEM)**
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

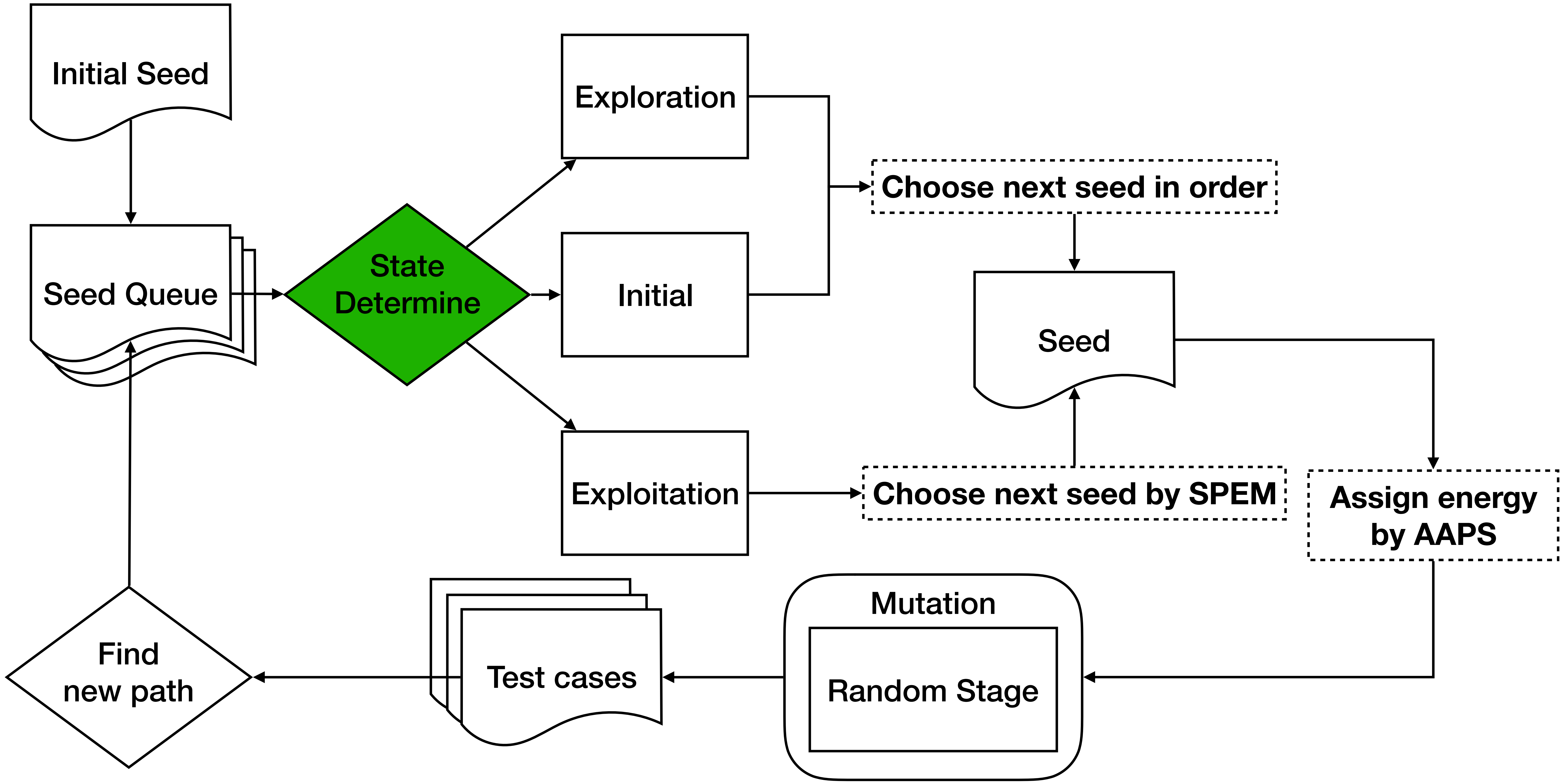
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: **Adaptive Average-cost-based Power Schedule (AAPS)**
- Three states of VAMAB

EcoFuzz

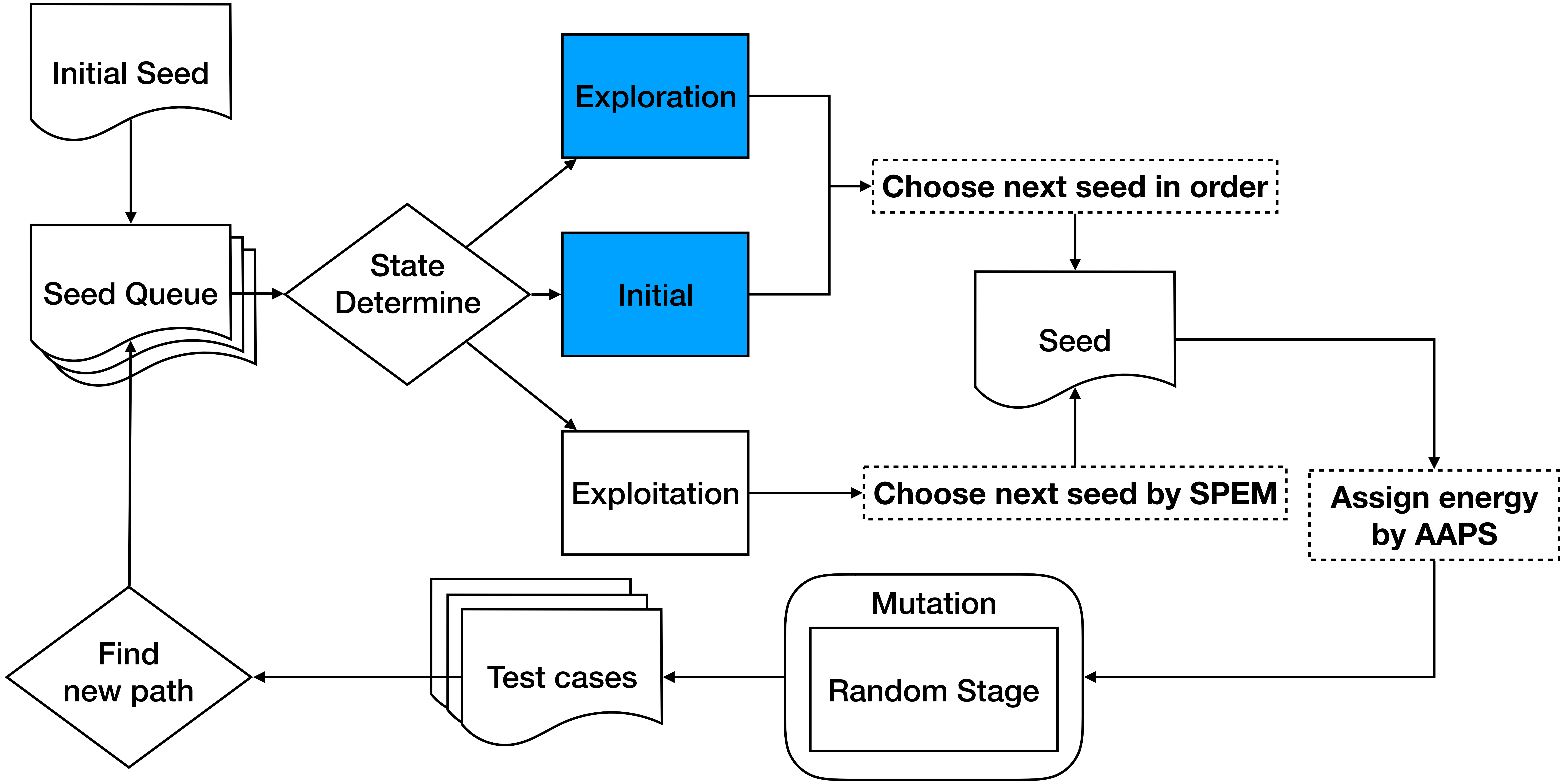
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

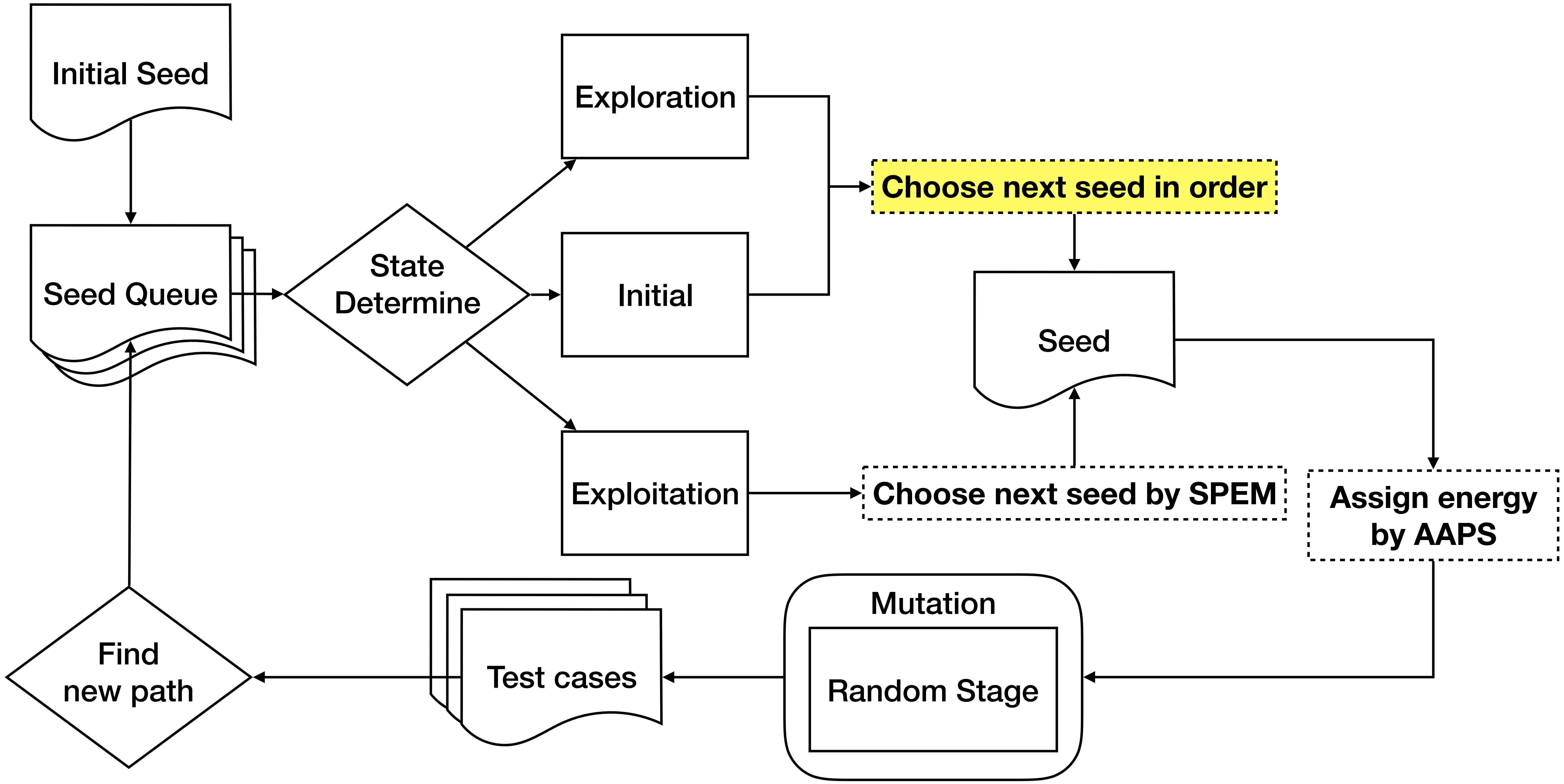
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

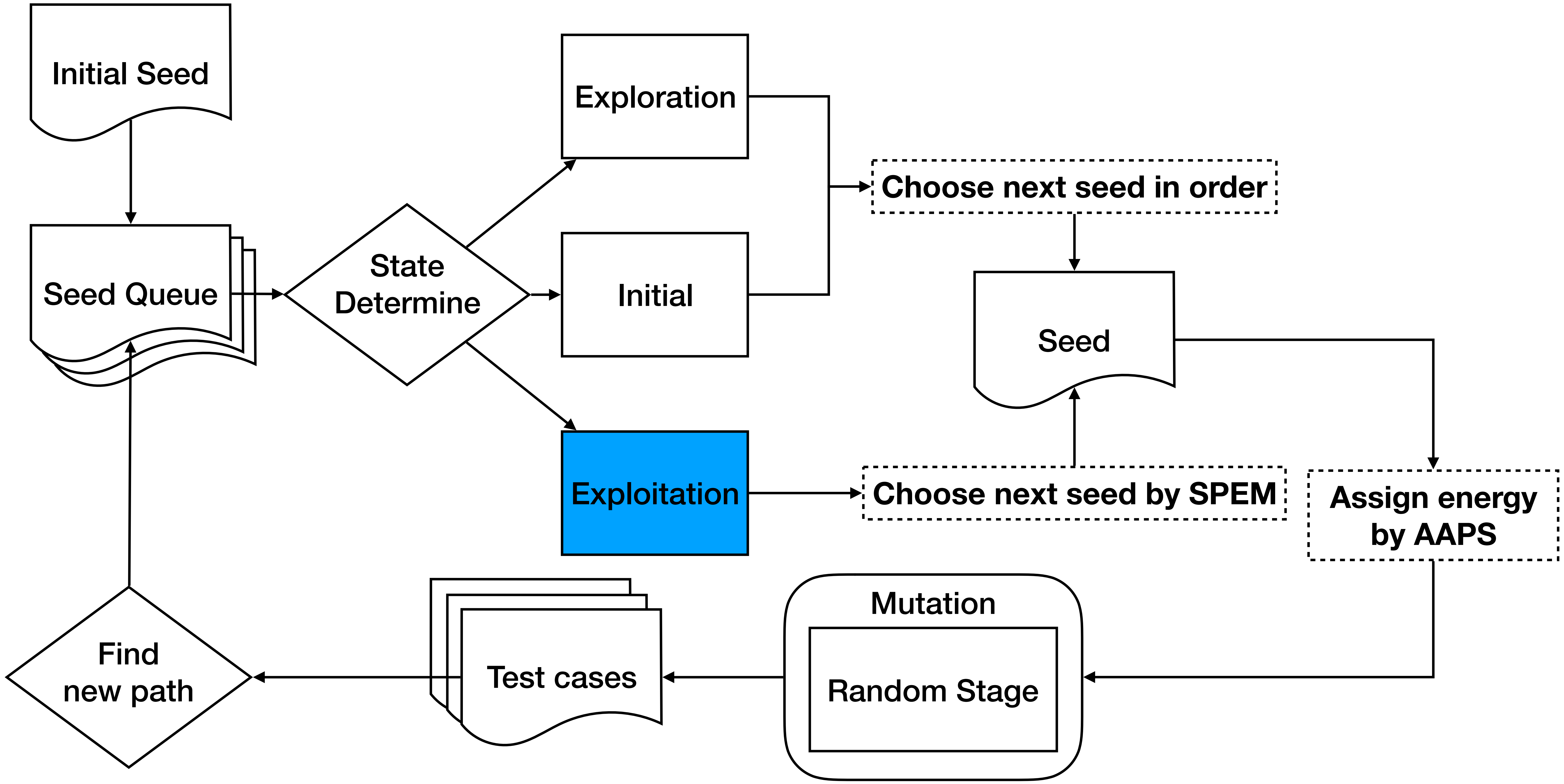
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

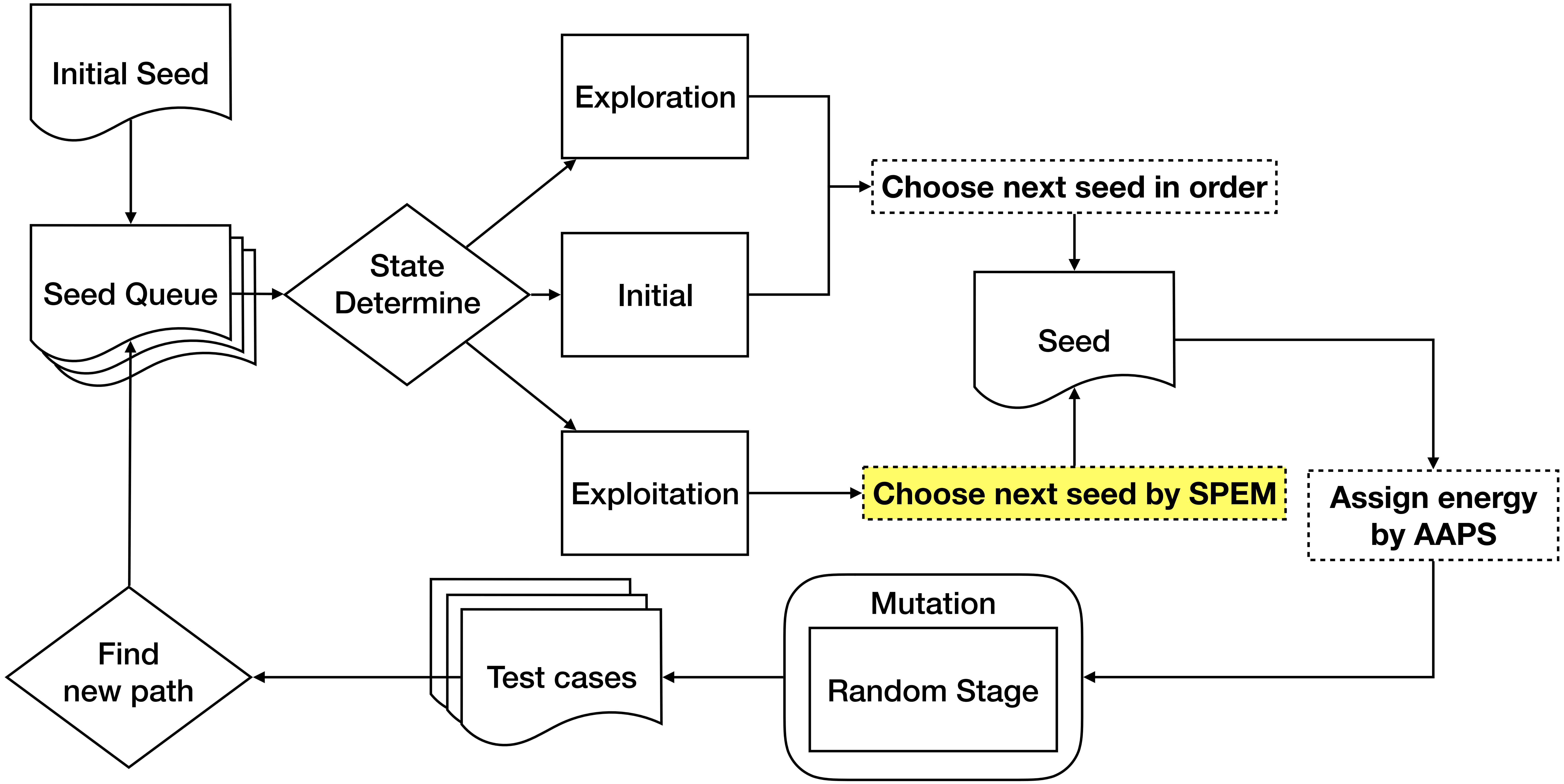
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

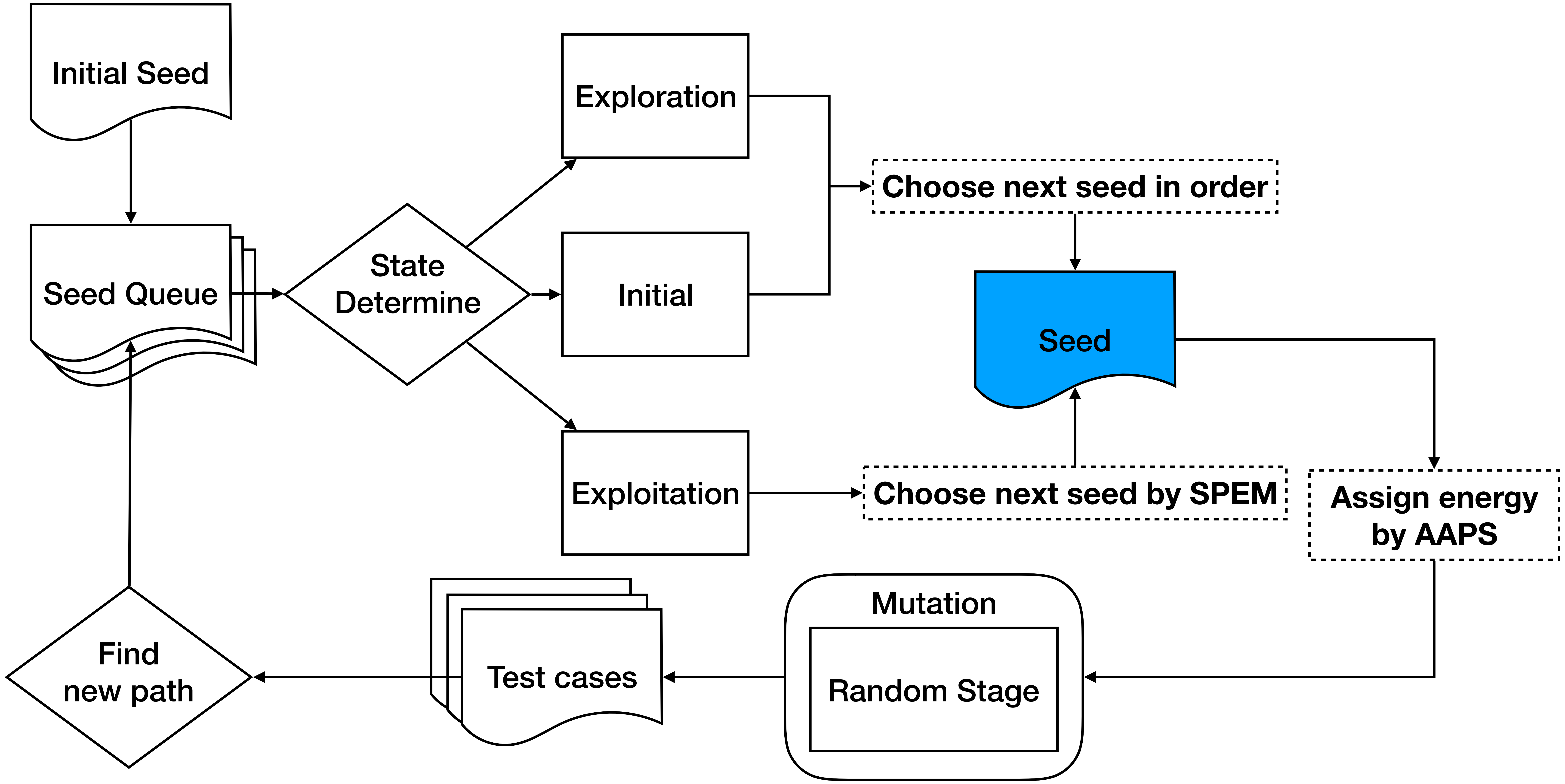
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

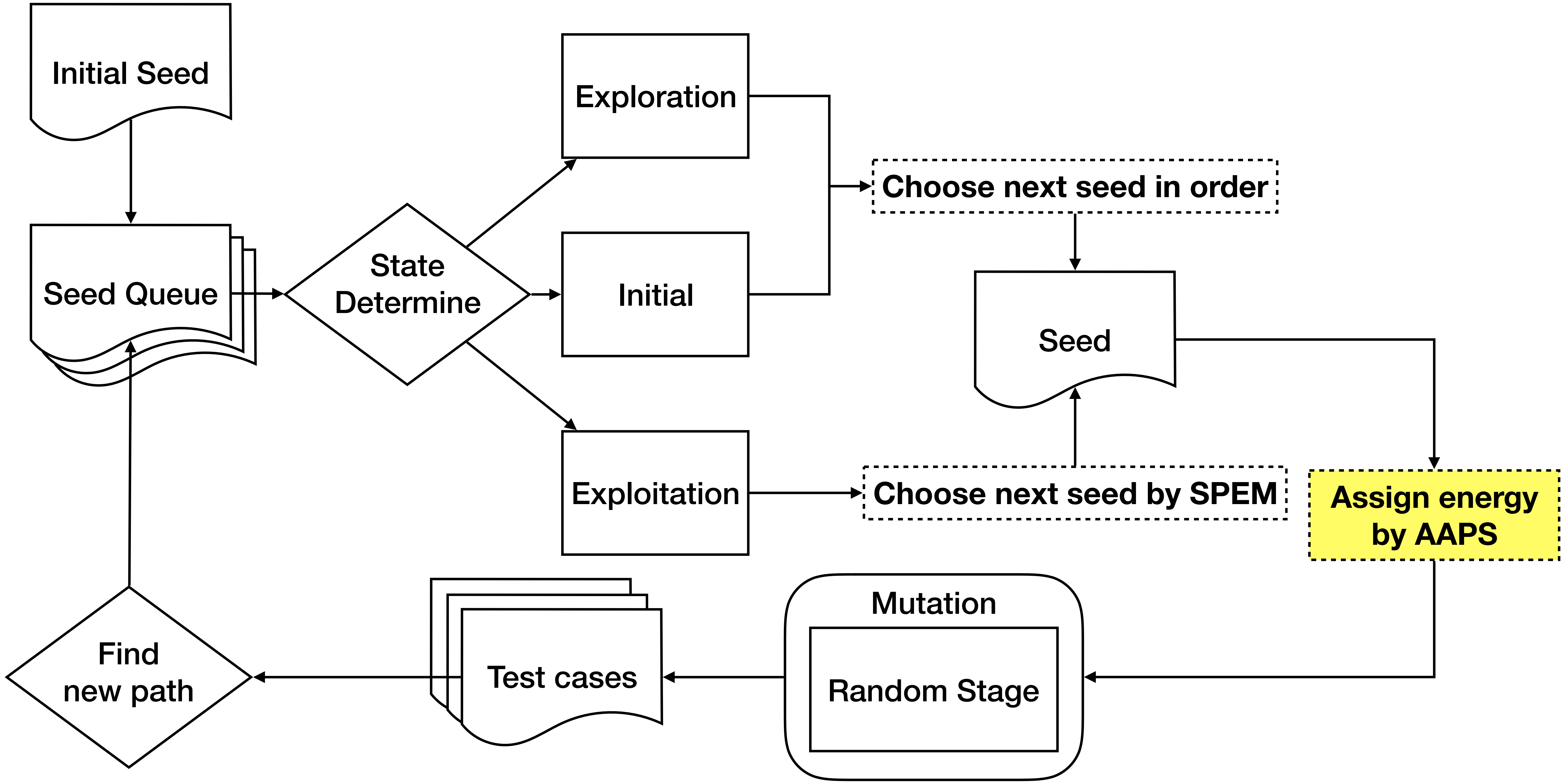
- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

EcoFuzz

- Main Framework



- Based on AFL
- Search strategy: Self-transition-based Probability Estimation Method (SPEM)
- Power schedule: Adaptive Average-cost-based Power Schedule (AAPS)
- Three states of VAMAB

Self-transition-based Probability Estimation Method (SPEM)

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy
  - estimate the **reward probabilities**
  - select the next seed in **exploitation state**

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy
  - estimate the **reward probabilities**
  - select the next seed in **exploitation state**
- Estimating probability by frequency

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} = 1 - p_{ii} - \sum_{j=1, j\neq i}^{n} p_{ij}$$

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy
  - estimate the **reward probabilities**
  - select the next seed in **exploitation state**
- Estimating probability by frequency

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} = 1 - \boxed{p_{ii}} - \sum_{j=1, j \neq i}^{n} p_{ij}$$

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy
  - estimate the **reward probabilities**
  - select the next seed in **exploitation state**
- Estimating probability by frequency
  - $p_{ii} \approx f_{ii}$

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} = 1 - \boxed{p_{ii}} - \sum_{j=1,j\neq i}^{n} p_{ij}$$

↓

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} \approx 1 - f_{ii} - \sum_{j=1,j\neq i}^{n} p_{ij}$$

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy
  - estimate the **reward probabilities**
  - select the next seed in **exploitation state**
- Estimating probability by frequency
  - $p_{ii} \approx f_{ii}$

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} = 1 - \boxed{p_{ii}} - \sum_{j=1, j \neq i}^{n} p_{ij}$$

$$\Big\downarrow$$

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} \approx 1 - \boxed{f_{ii} - \sum_{j=1, j \neq i}^{n} p_{ij}}$$

Self-transition-based Probability Estimation Method (SPEM)

- Search strategy
  - estimate the **reward probabilities**
  - select the next seed in **exploitation state**
- Estimating probability by frequency
  - $p_{ii} \approx f_{ii}$
  - $f_{ii} + \sum_{j=1,j\neq i}^{n} p_{ij} \approx \frac{f_{ii}}{\sqrt{i}}$

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} = 1 - \boxed{p_{ii}} - \sum_{j=1,j\neq i}^{n} p_{ij}$$

$$\Big\downarrow$$

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} \approx 1 - \boxed{f_{ii} - \sum_{j=1,j\neq i}^{n} p_{ij}}$$

$$\Big\downarrow$$

$$P_{R_{i,n}} = 1 - \sum_{j=1}^{n} p_{ij} \approx 1 - \boxed{\frac{f_{ii}}{\sqrt{i}}}$$

Adaptive Average-cost-based Power Schedule (AAPS)

Adaptive Average-cost-based Power Schedule (AAPS)

- **Average-cost**

Adaptive Average-cost-based Power Schedule (AAPS)

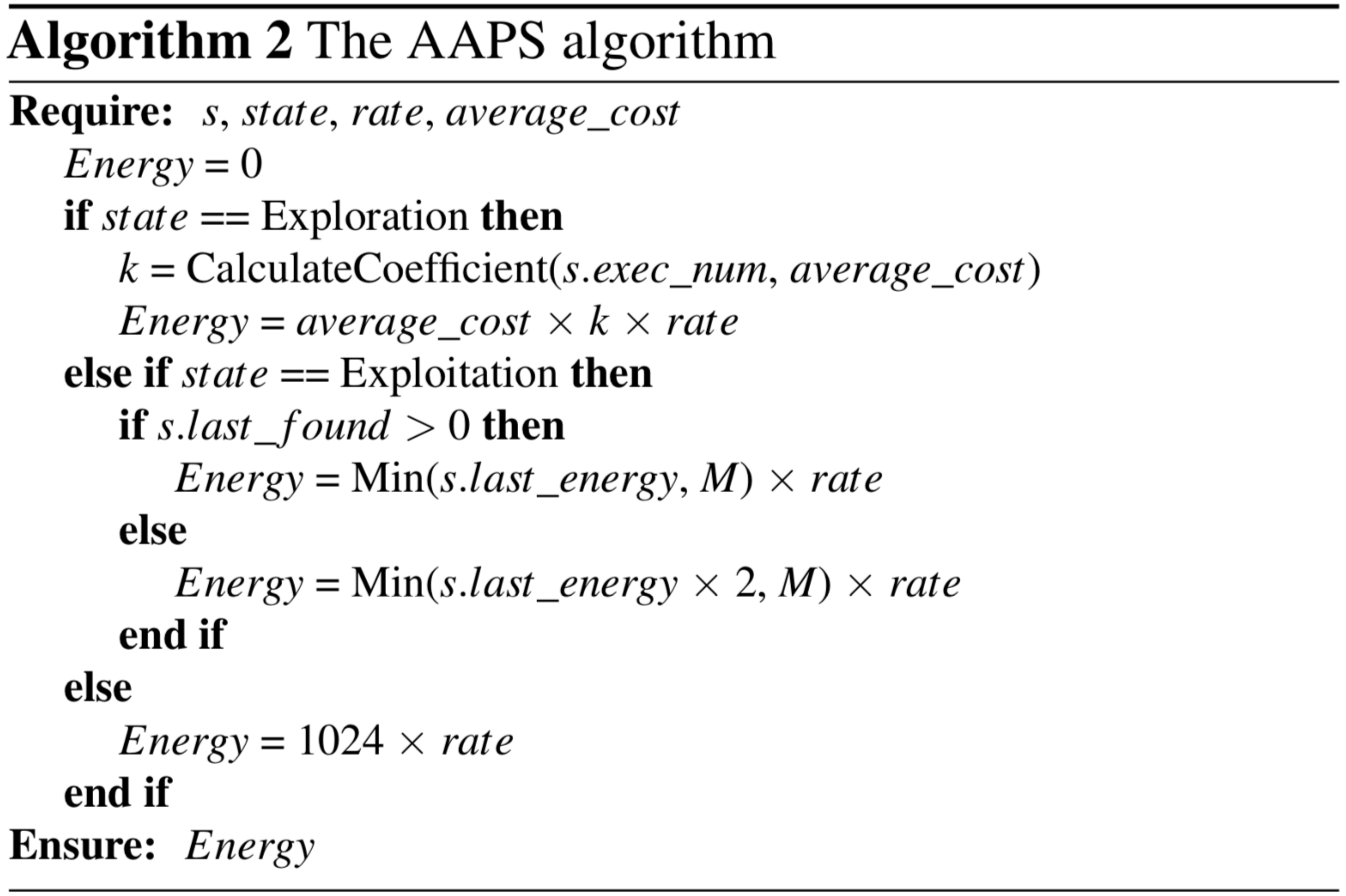
- **Average-cost**
  - $C = \dfrac{total\_testcases}{found\_paths}$

Adaptive Average-cost-based Power Schedule (AAPS)

- **Average-cost**
  - $$C = \frac{total\_testcases}{found\_paths}$$
- Utilize **average-cost** as the **basic line**

Adaptive Average-cost-based Power Schedule (AAPS)

- **Average-cost**
  - $$C = \frac{total\_testcases}{found\_paths}$$
- Utilize **average-cost** as the **basic line**

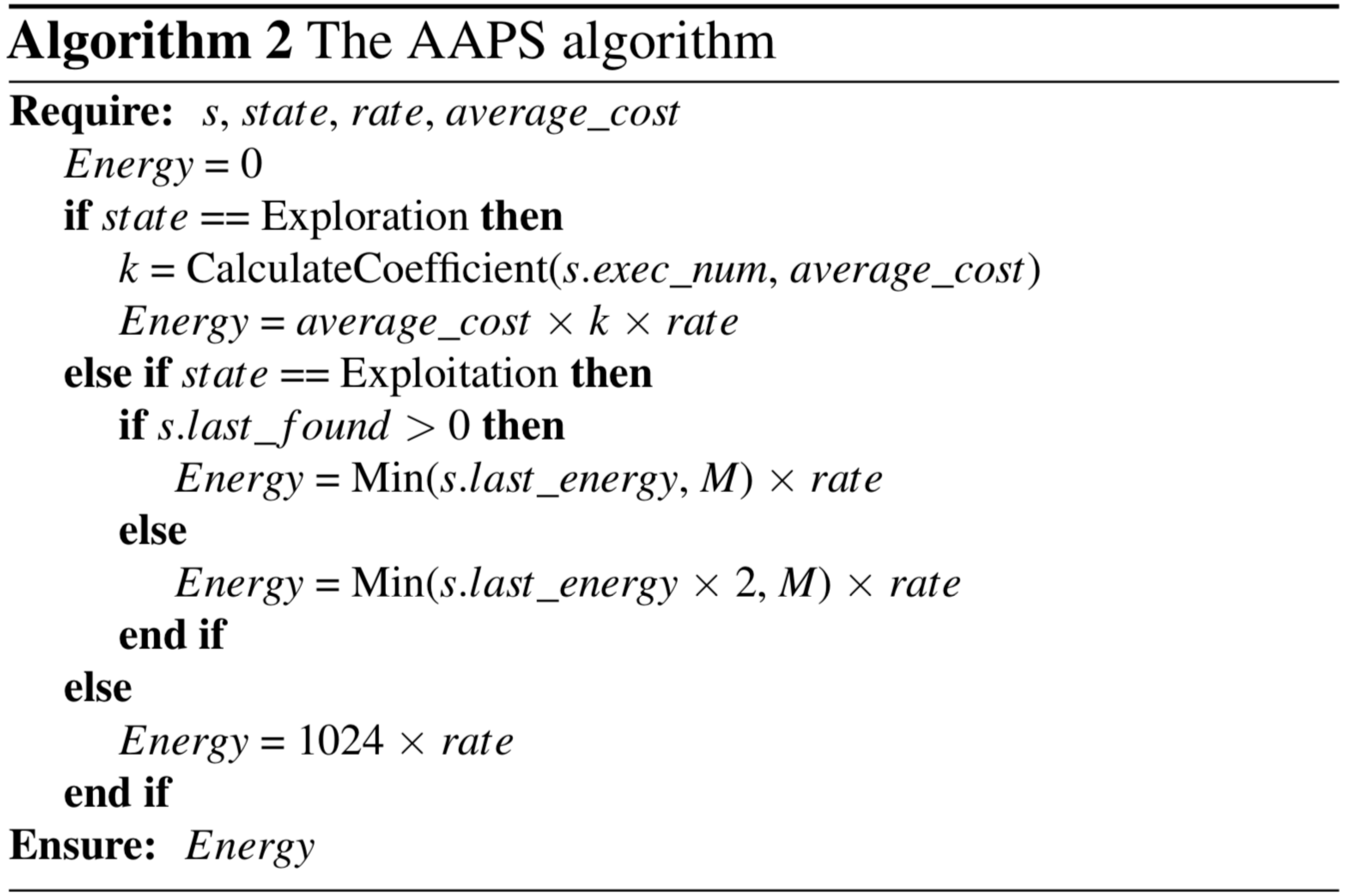


Algorithm 2 The AAPS algorithm

```
Require: $s$, $state$, $rate$, $average\_cost$
    $Energy = 0$
    if $state$ == Exploration then
        $k$ = CalculateCoefficient($s.exec\_num$, $average\_cost$)
        $Energy = average\_cost \times k \times rate$
    else if $state$ == Exploitation then
        if $s.last\_found > 0$ then
            $Energy$ = Min($s.last\_energy$, $M$) $\times$ $rate$
        else
            $Energy$ = Min($s.last\_energy \times 2$, $M$) $\times$ $rate$
        end if
    else
        $Energy = 1024 \times rate$
    end if
Ensure: $Energy$
```

Adaptive Average-cost-based Power Schedule (AAPS)

- **Average-cost**
  - $C = \dfrac{total\_testcases}{found\_paths}$
- Utilize **average-cost** as the **basic line**
  - Allocating energy no more than average-cost in the **exploration stage**

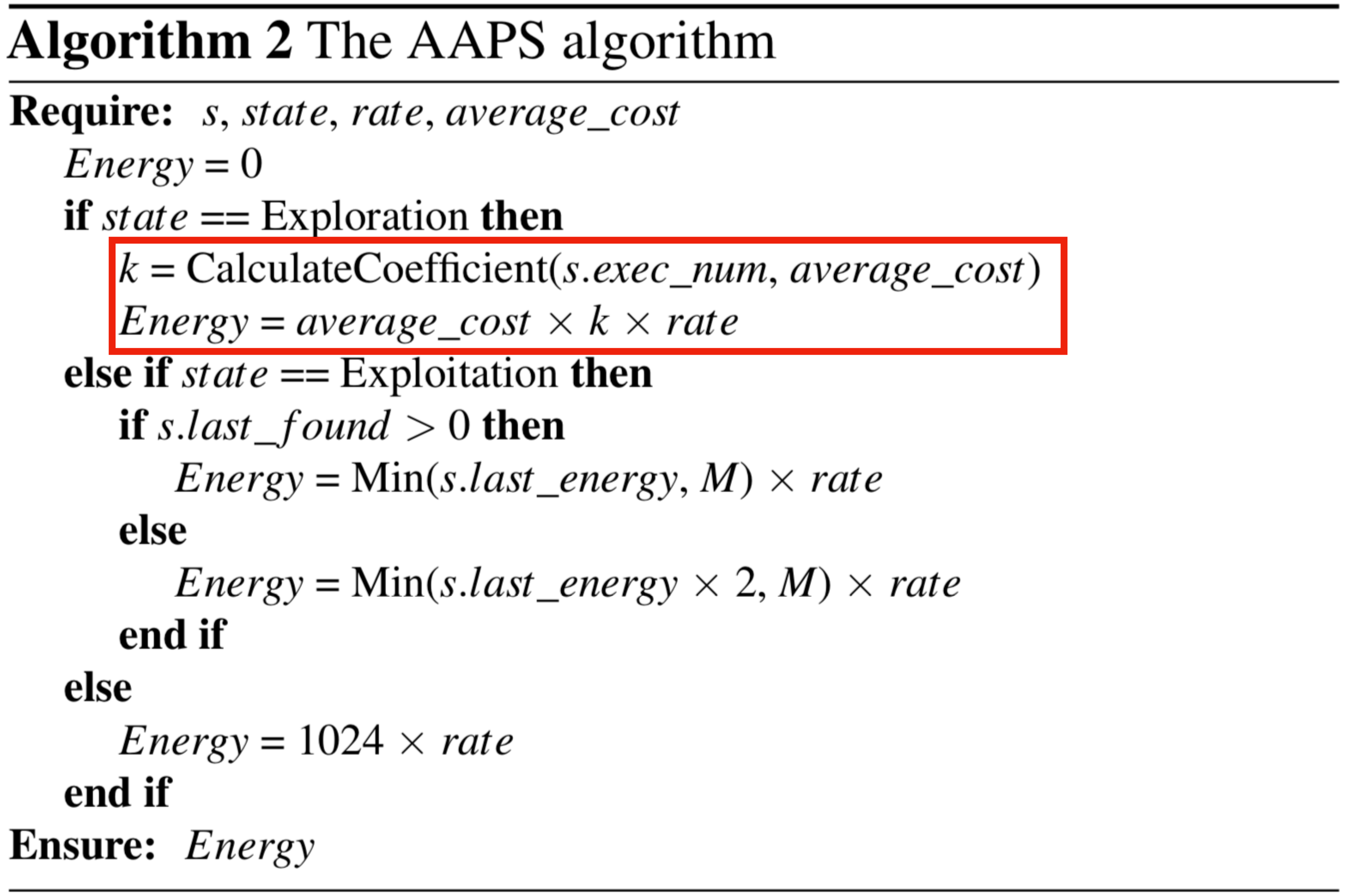


Algorithm 2 The AAPS algorithm

```
Require: $s$, $state$, $rate$, $average\_cost$
  $Energy = 0$
  if $state$ == Exploration then
    $k$ = CalculateCoefficient($s.exec\_num$, $average\_cost$)
    $Energy = average\_cost \times k \times rate$
  else if $state$ == Exploitation then
    if $s.last\_found > 0$ then
      $Energy = \mathrm{Min}(s.last\_energy, M) \times rate$
    else
      $Energy = \mathrm{Min}(s.last\_energy \times 2, M) \times rate$
    end if
  else
    $Energy = 1024 \times rate$
  end if
Ensure: $Energy$
```

Adaptive Average-cost-based Power Schedule (AAPS)

- **Average-cost**
  - $$C = \frac{total\_testcases}{found\_paths}$$
- Utilize **average-cost** as the **basic line**
  - Allocating energy no more than average-cost in the **exploration stage**
  - Less energy allocated on seeds exercising **high-frequency paths**



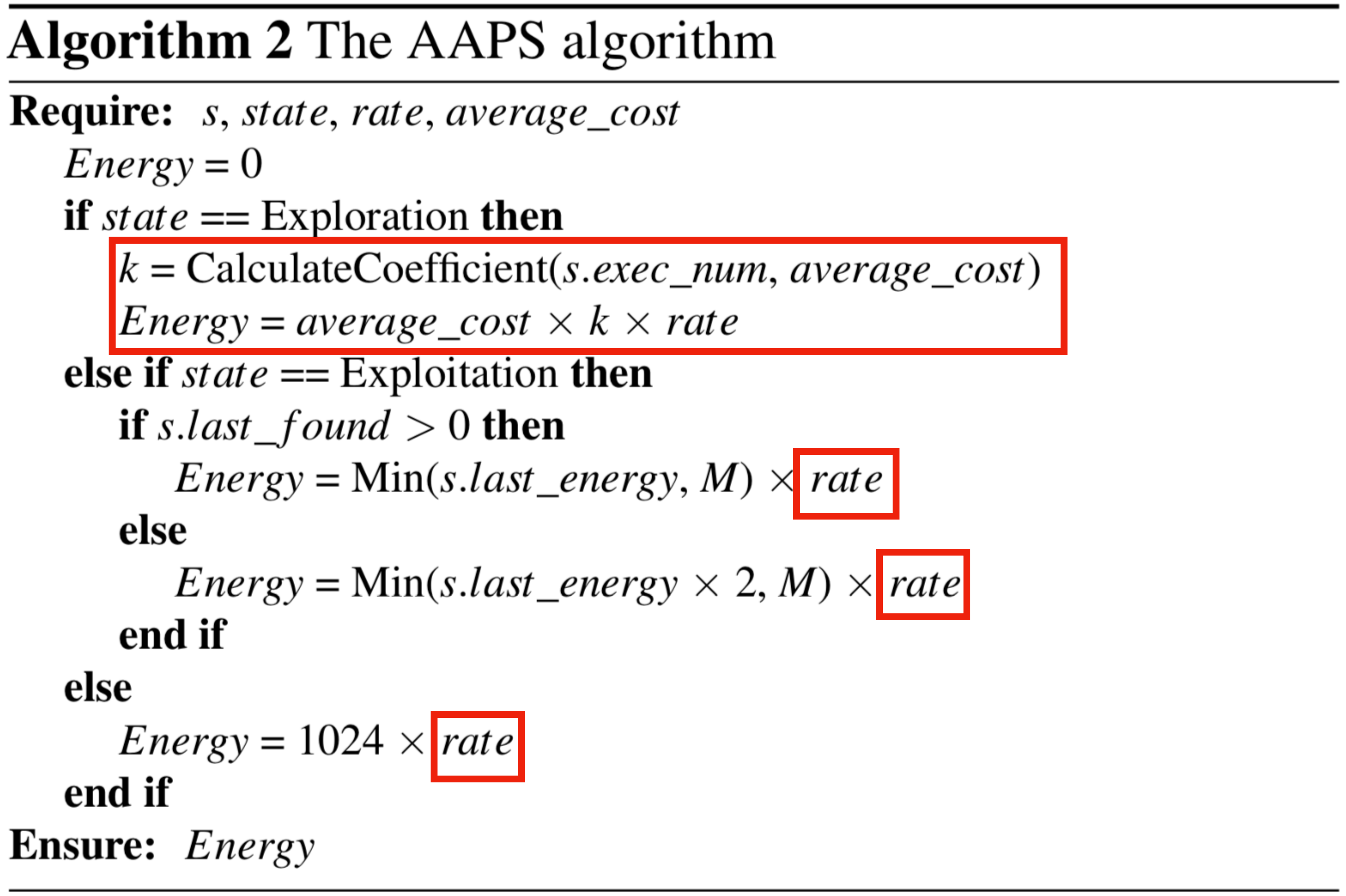
Algorithm 2 The AAPS algorithm

```
Require: $s$, $state$, $rate$, $average\_cost$
  $Energy = 0$
  if $state$ == Exploration then
    $k$ = CalculateCoefficient($s.exec\_num$, $average\_cost$)
    $Energy = average\_cost \times k \times rate$
  else if $state$ == Exploitation then
    if $s.last\_found > 0$ then
      $Energy = \mathrm{Min}(s.last\_energy, M) \times rate$
    else
      $Energy = \mathrm{Min}(s.last\_energy \times 2, M) \times rate$
    end if
  else
    $Energy = 1024 \times rate$
  end if
Ensure: $Energy$
```

Adaptive Average-cost-based Power Schedule (AAPS)

- **Average-cost**
  - $C = \dfrac{total\_testcases}{found\_paths}$

- Utilize **average-cost** as the **basic line**
  - Allocating energy no more than average-cost in the **exploration stage**
  - Less energy allocated on seeds exercising **high-frequency paths**
  - A **context-adaptive** energy allocation mechanism



Algorithm 2 The AAPS algorithm

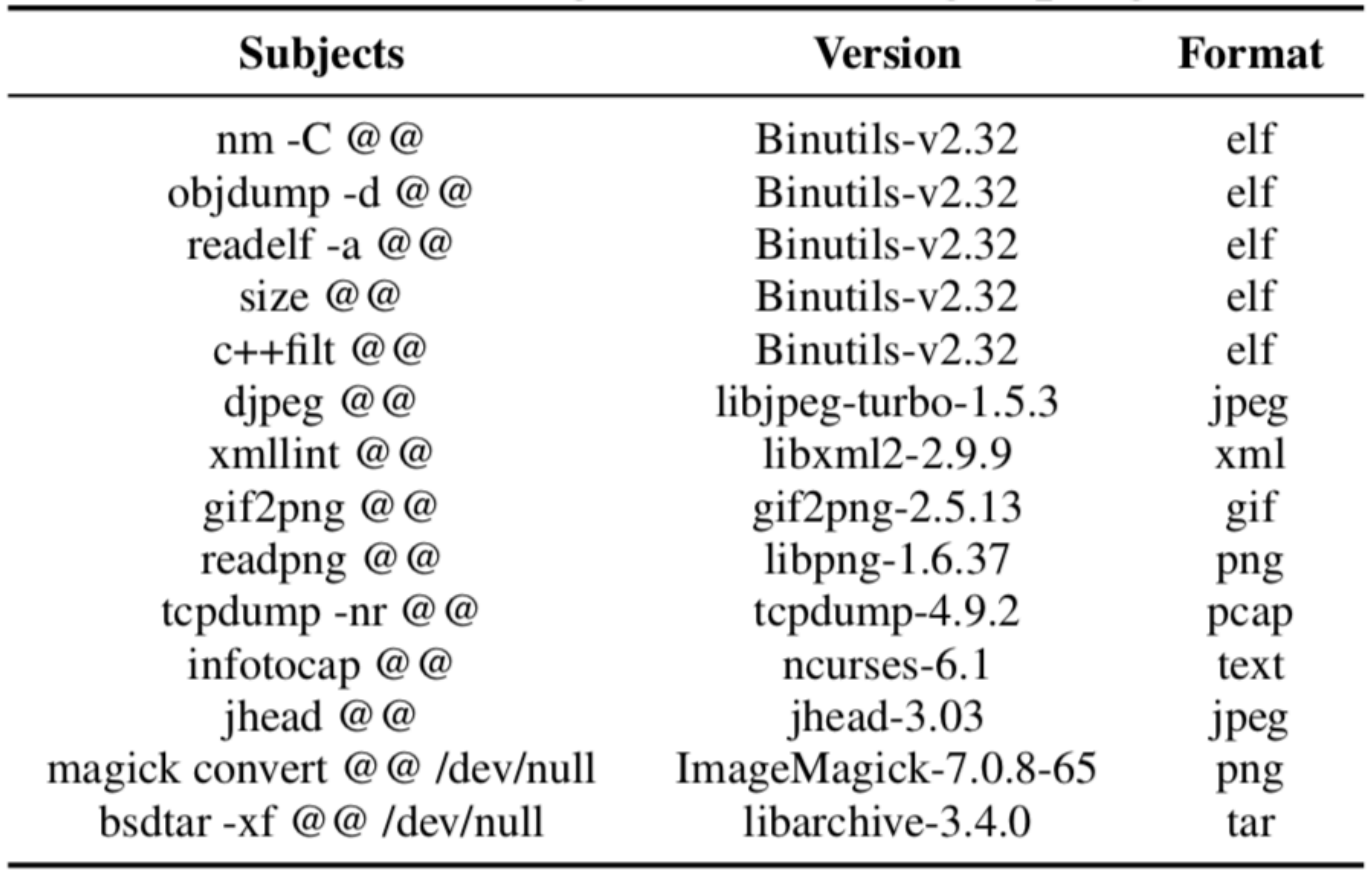
```
Require: $s$, $state$, $rate$, $average\_cost$
  $Energy = 0$
  if $state$ == Exploration then
    $k$ = CalculateCoefficient($s.exec\_num$, $average\_cost$)
    $Energy = average\_cost \times k \times rate$
  else if $state$ == Exploitation then
    if $s.last\_found > 0$ then
      $Energy$ = Min($s.last\_energy$, $M$) $\times$ $rate$
    else
      $Energy$ = Min($s.last\_energy \times 2$, $M$) $\times$ $rate$
    end if
  else
    $Energy = 1024 \times rate$
  end if
Ensure: $Energy$
```

Contributions

- One model: a variant of the Adversarial Multi-Armed Bandit (VAMAB)
- One tool: an adaptive energy-saving fuzzer named EcoFuzz
- Comprehensive evaluation: a serial of experiments from different metrics

Evaluation

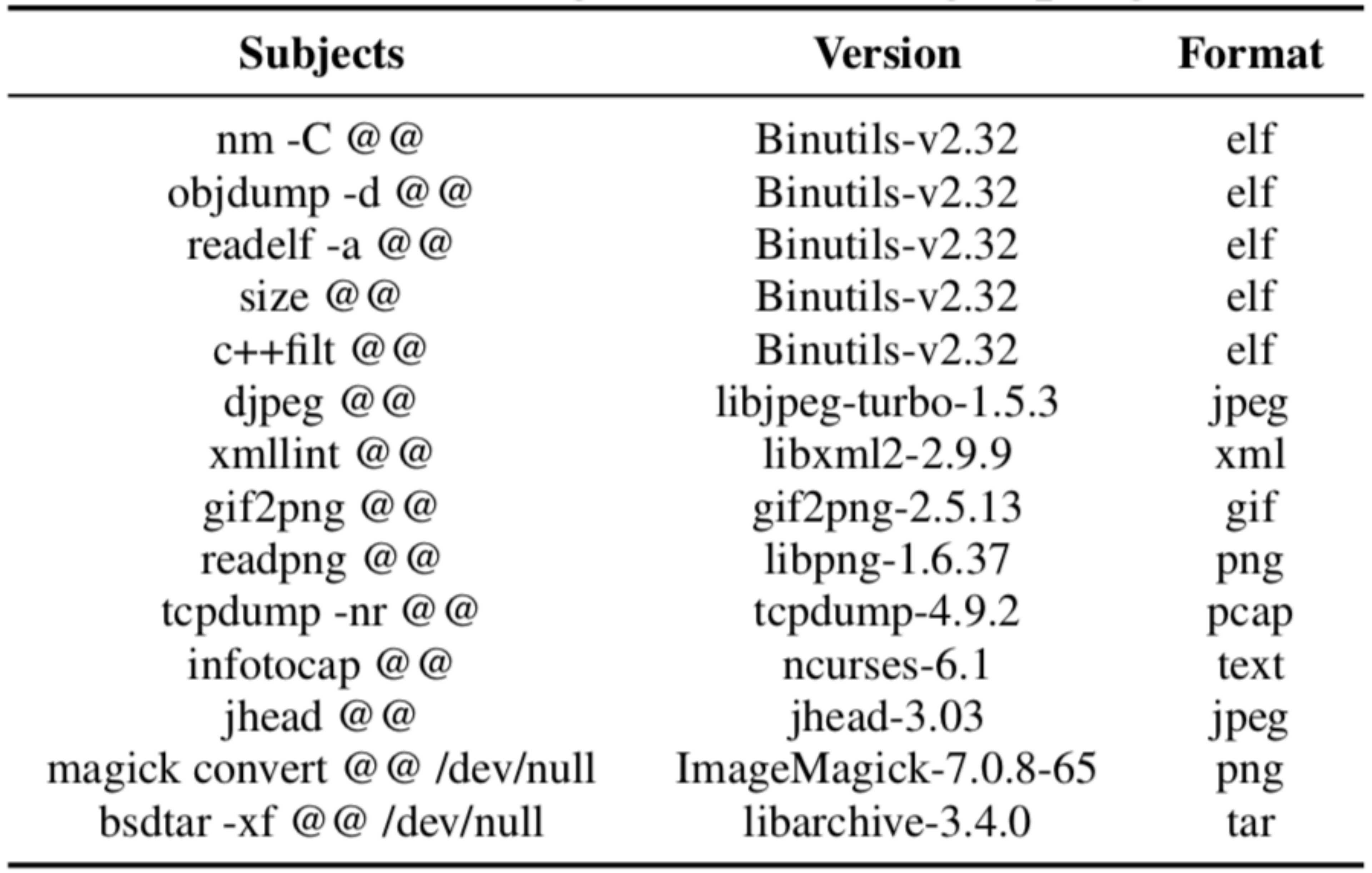
- 14 real-world programs
- Compared with 7 state-of-the-art tools
  - AFL, AFLFast, FidgetyAFL, AFLFast.new, MOPT, FairFuzz
- Configuration:
  - 24 hours with 5 times
- Evaluation metric:
  - The number of discovered paths
  - The number of generated test cases
  - Average-cost



Subjects	Version	Format
nm -C @@	Binutils-v2.32	elf
objdump -d @@	Binutils-v2.32	elf
readelf -a @@	Binutils-v2.32	elf
size @@	Binutils-v2.32	elf
c++filt @@	Binutils-v2.32	elf
djpeg @@	libjpeg-turbo-1.5.3	jpeg
xmllint @@	libxml2-2.9.9	xml
gif2png @@	gif2png-2.5.13	gif
readpng @@	libpng-1.6.37	png
tcpdump -nr @@	tcpdump-4.9.2	pcap
infotocap @@	ncurses-6.1	text
jhead @@	jhead-3.03	jpeg
magick convert @@ /dev/null	ImageMagick-7.0.8-65	png
bsdtar -xf @@ /dev/null	libarchive-3.4.0	tar

Evaluation

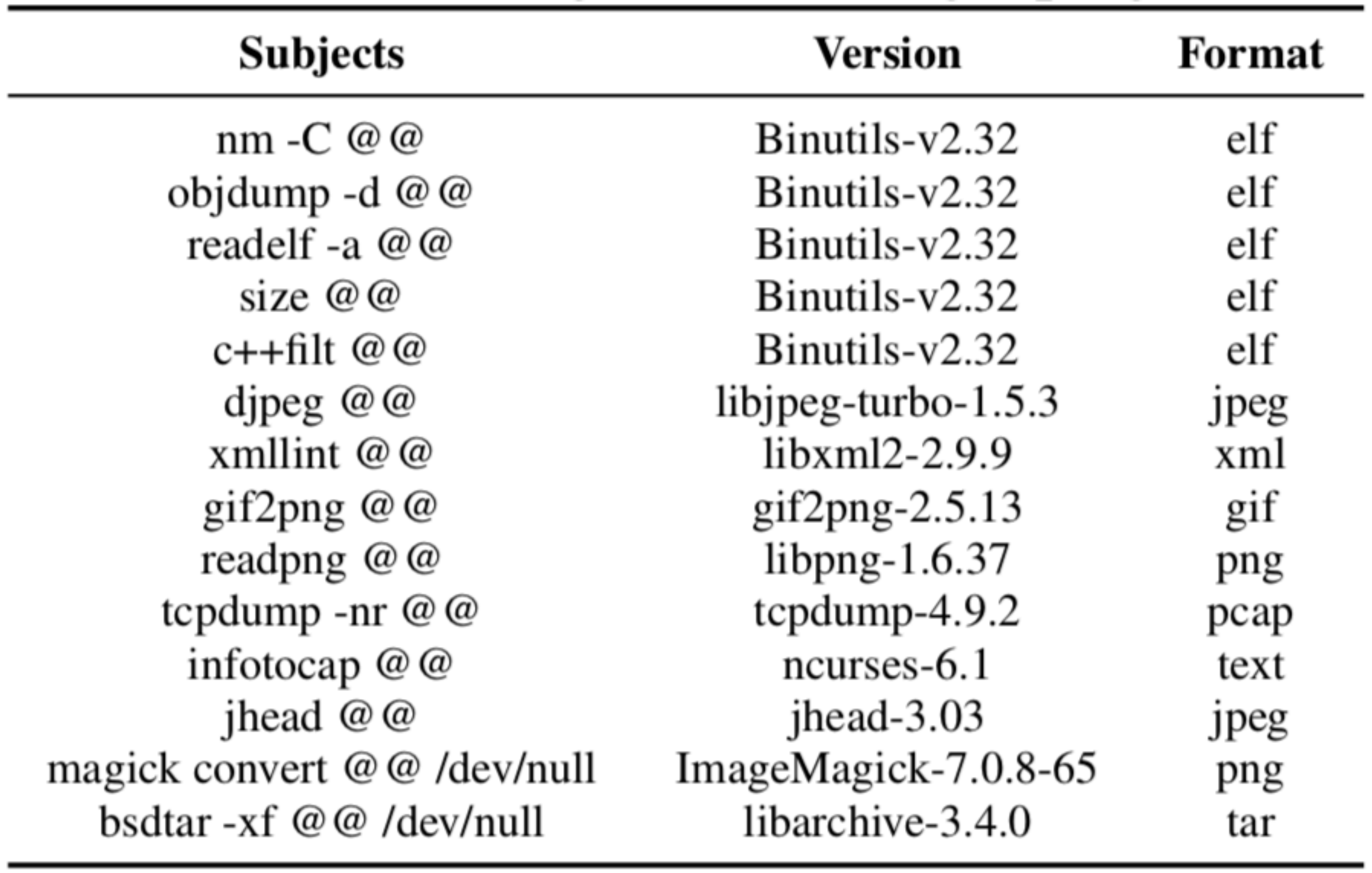
- 14 real-world programs
- Compared with **7 state-of-the-art** tools
    - AFL, AFLFast, FidgetyAFL, AFLFast.new, MOPT, FairFuzz
- Configuration:
    - 24 hours with 5 times
- Evaluation metric:
    - The number of discovered paths
    - The number of generated test cases
    - Average-cost



Subjects	**Version**	**Format**
nm -C @@	Binutils-v2.32	elf
objdump -d @@	Binutils-v2.32	elf
readelf -a @@	Binutils-v2.32	elf
size @@	Binutils-v2.32	elf
c++filt @@	Binutils-v2.32	elf
djpeg @@	libjpeg-turbo-1.5.3	jpeg
xmllint @@	libxml2-2.9.9	xml
gif2png @@	gif2png-2.5.13	gif
readpng @@	libpng-1.6.37	png
tcpdump -nr @@	tcpdump-4.9.2	pcap
infotocap @@	ncurses-6.1	text
jhead @@	jhead-3.03	jpeg
magick convert @@ /dev/null	ImageMagick-7.0.8-65	png
bsdtar -xf @@ /dev/null	libarchive-3.4.0	tar

Evaluation

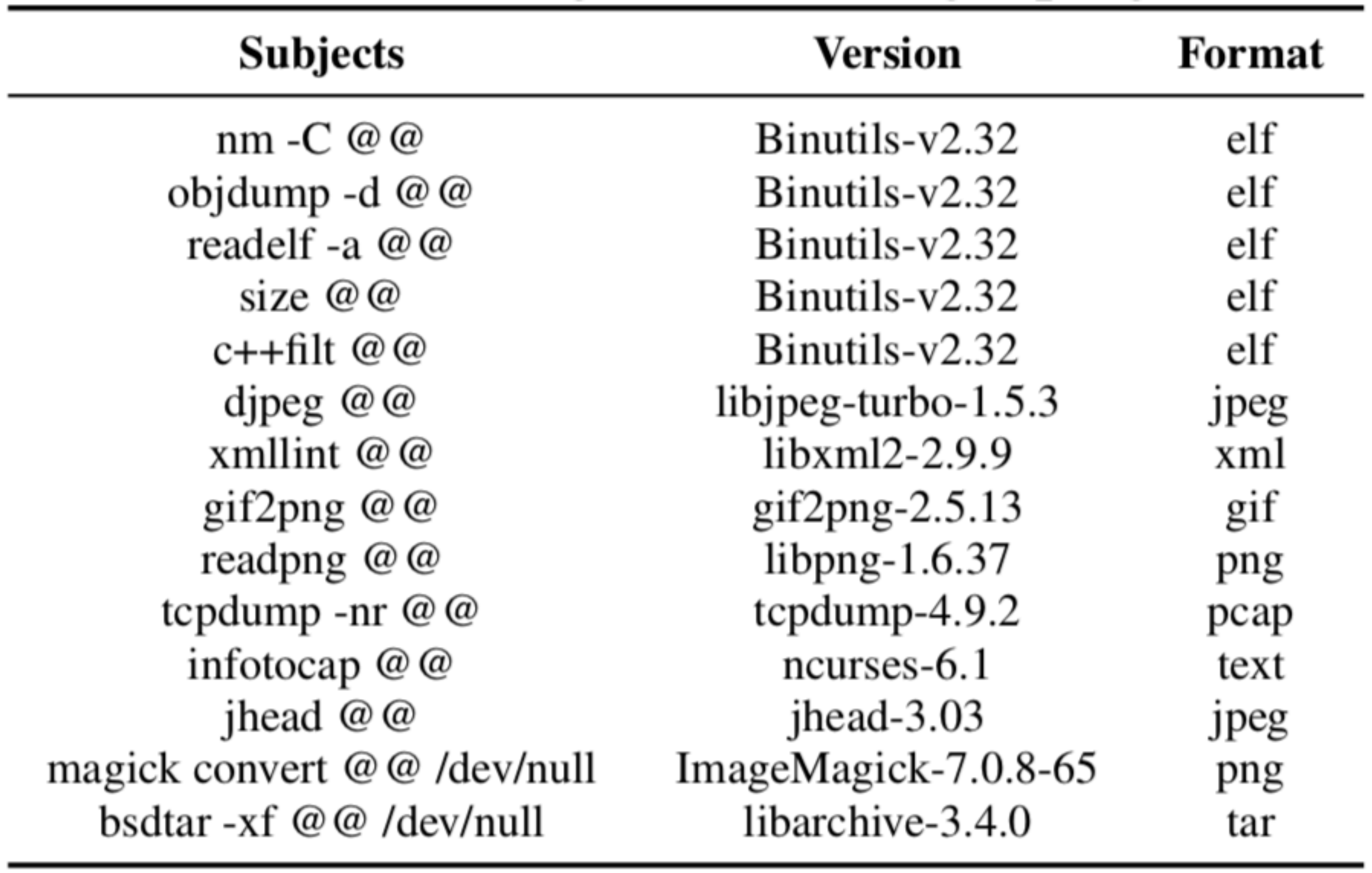
- 14 real-world programs
- Compared with 7 state-of-the-art tools
  - AFL, AFLFast, FidgetyAFL, AFLFast.new, MOPT, FairFuzz
- Configuration:
  - 24 hours with 5 times
- Evaluation metric:
  - The number of discovered paths
  - The number of generated test cases
  - Average-cost



Subjects	Version	Format
nm -C @@	Binutils-v2.32	elf
objdump -d @@	Binutils-v2.32	elf
readelf -a @@	Binutils-v2.32	elf
size @@	Binutils-v2.32	elf
c++filt @@	Binutils-v2.32	elf
djpeg @@	libjpeg-turbo-1.5.3	jpeg
xmllint @@	libxml2-2.9.9	xml
gif2png @@	gif2png-2.5.13	gif
readpng @@	libpng-1.6.37	png
tcpdump -nr @@	tcpdump-4.9.2	pcap
infotocap @@	ncurses-6.1	text
jhead @@	jhead-3.03	jpeg
magick convert @@ /dev/null	ImageMagick-7.0.8-65	png
bsdtar -xf @@ /dev/null	libarchive-3.4.0	tar

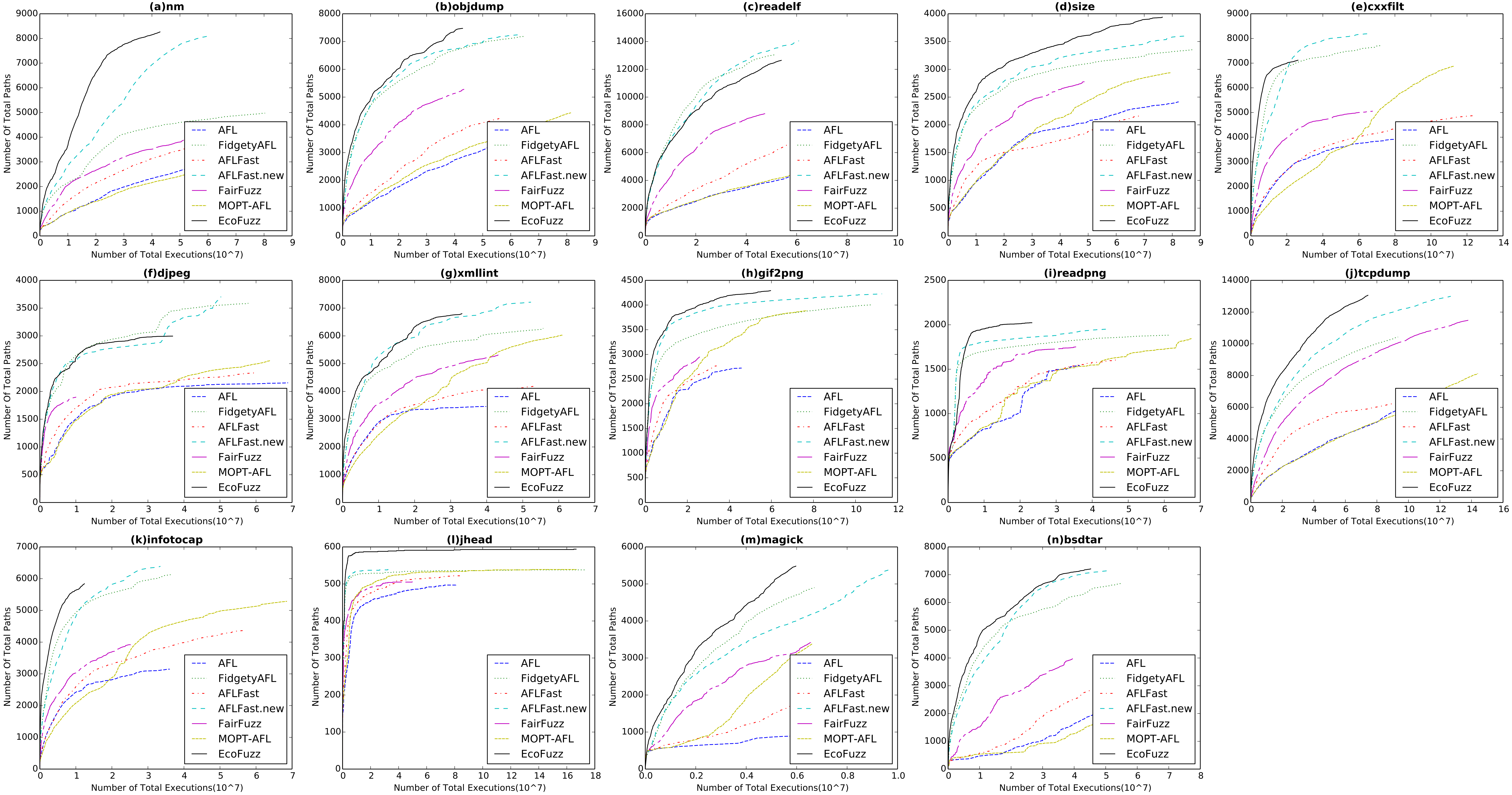
Evaluation

- 14 real-world programs
- Compared with 7 state-of-the-art tools
  - AFL, AFLFast, FidgetyAFL, AFLFast.new, MOPT, FairFuzz
- Configuration:
  - 24 hours with 5 times
- Evaluation metric:
  - The number of discovered paths
  - The number of generated test cases
  - Average-cost

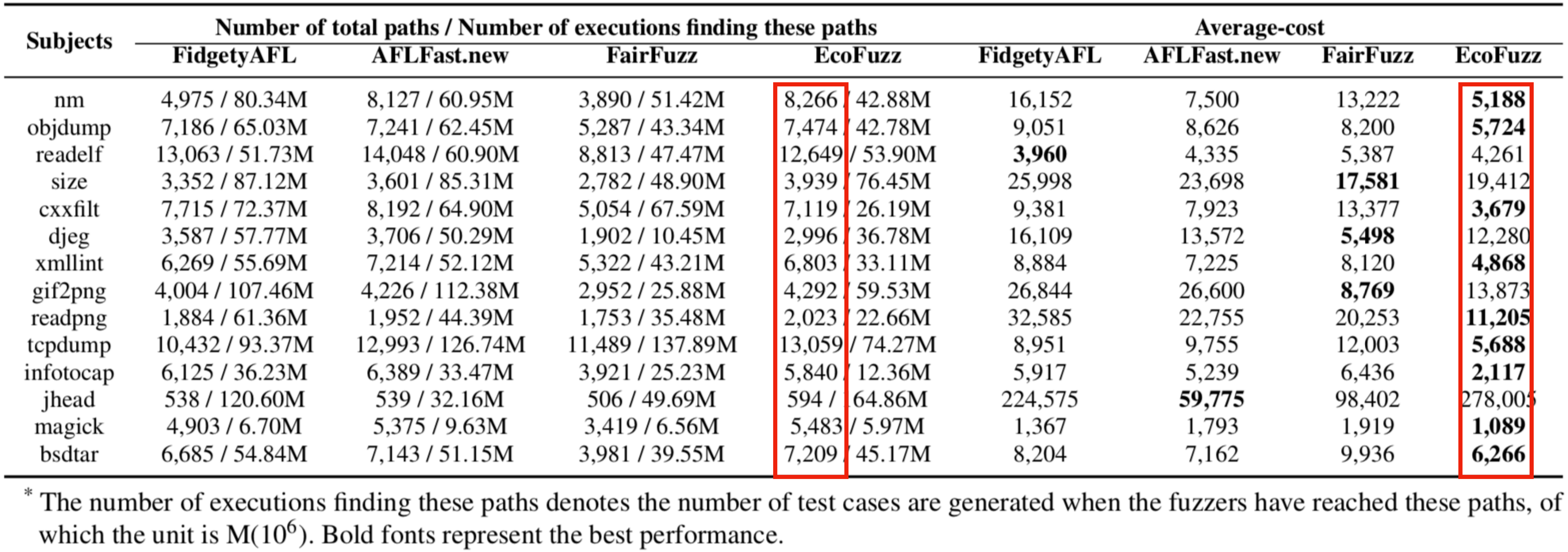


Subjects	Version	Format
nm -C @@	Binutils-v2.32	elf
objdump -d @@	Binutils-v2.32	elf
readelf -a @@	Binutils-v2.32	elf
size @@	Binutils-v2.32	elf
c++filt @@	Binutils-v2.32	elf
djpeg @@	libjpeg-turbo-1.5.3	jpeg
xmllint @@	libxml2-2.9.9	xml
gif2png @@	gif2png-2.5.13	gif
readpng @@	libpng-1.6.37	png
tcpdump -nr @@	tcpdump-4.9.2	pcap
infotocap @@	ncurses-6.1	text
jhead @@	jhead-3.03	jpeg
magick convert @@ /dev/null	ImageMagick-7.0.8-65	png
bsdtar -xf @@ /dev/null	libarchive-3.4.0	tar

Evaluation


(a)nm
(b)objdump
(c)readelf
(d)size
(e)cxxfilt
(f)djpeg
(g)xmllint
(h)gif2png
(i)readpng
(j)tcpdump
(k)infotocap
(l)jhead
(m)magick
(n)bsdtar
Number Of Total Paths
Number of Total Executions(10^7)
AFL
FidgetyAFL
AFLFast
AFLFast.new
FairFuzz
MOPT-AFL
EcoFuzz

Evaluation



Subjects	Number of total paths / Number of executions finding these paths				Average-cost			
	FidgetyAFL	AFLFast.new	FairFuzz	EcoFuzz	FidgetyAFL	AFLFast.new	FairFuzz	EcoFuzz
nm	4,975 / 80.34M	8,127 / 60.95M	3,890 / 51.42M	8,266 / 42.88M	16,152	7,500	13,222	**5,188**
objdump	7,186 / 65.03M	7,241 / 62.45M	5,287 / 43.34M	7,474 / 42.78M	9,051	8,626	8,200	**5,724**
readelf	13,063 / 51.73M	14,048 / 60.90M	8,813 / 47.47M	12,649 / 53.90M	**3,960**	4,335	5,387	4,261
size	3,352 / 87.12M	3,601 / 85.31M	2,782 / 48.90M	3,939 / 76.45M	25,998	23,698	**17,581**	19,412
cxxfilt	7,715 / 72.37M	8,192 / 64.90M	5,054 / 67.59M	7,119 / 26.19M	9,381	7,923	13,377	**3,679**
djeg	3,587 / 57.77M	3,706 / 50.29M	1,902 / 10.45M	2,996 / 36.78M	16,109	13,572	**5,498**	12,280
xmllint	6,269 / 55.69M	7,214 / 52.12M	5,322 / 43.21M	6,803 / 33.11M	8,884	7,225	8,120	**4,868**
gif2png	4,004 / 107.46M	4,226 / 112.38M	2,952 / 25.88M	4,292 / 59.53M	26,844	26,600	**8,769**	13,873
readpng	1,884 / 61.36M	1,952 / 44.39M	1,753 / 35.48M	2,023 / 22.66M	32,585	22,755	20,253	**11,205**
tcpdump	10,432 / 93.37M	12,993 / 126.74M	11,489 / 137.89M	13,059 / 74.27M	8,951	9,755	12,003	**5,688**
infotocap	6,125 / 36.23M	6,389 / 33.47M	3,921 / 25.23M	5,840 / 12.36M	5,917	5,239	6,436	**2,117**
jhead	538 / 120.60M	539 / 32.16M	506 / 49.69M	594 / 64.86M	224,575	**59,775**	98,402	278,005
magick	4,903 / 6.70M	5,375 / 9.63M	3,419 / 6.56M	5,483 / 5.97M	1,367	1,793	1,919	**1,089**
bsdtar	6,685 / 54.84M	7,143 / 51.15M	3,981 / 39.55M	7,209 / 45.17M	8,204	7,162	9,936	**6,266**

* The number of executions finding these paths denotes the number of test cases are generated when the fuzzers have reached these paths, of which the unit is M(10^6). Bold fonts represent the best performance.

- Outperform other AFL-type techniques
  - EcoFuzz finds **214%** of the paths discovered by AFL and generates only **68%** test cases of AFL, while reducing **65%** average-cost of AFL

Evaluation

- Evaluate the efficiency of SPEM and AAPS

- Configuration:

  - choosing each best performance of EcoFuzz, FidgetyAFL, FairFuzz, and AFLFast.new on fuzzing **nm**

  - recording the energy allocated in random strategies of each turns, denoted as E_i, which i is the order of turn $(1 \leq i \leq N)$

  - recording the consumed energy for discovering the newest path of each turns, denoted as e_i, $0 \leq e_i \leq E_i$

  - recording the frequency of allocation with finding new paths for the seeds chosen repeatedly in the exploitation stage

Evaluation

- Evaluate the efficiency of SPEM and AAPS
- Configuration:
  - choosing each best performance of **EcoFuzz**, **FidgetyAFL**, **FairFuzz**, and **AFLFast.new** on fuzzing **nm**
  - recording the energy allocated in random strategies of each turns, denoted as E_i, which i is the order of turn $(1 \leq i \leq N)$
  - recording the consumed energy for discovering the newest path of each turns, denoted as e_i, $0 \leq e_i \leq E_i$
  - recording the frequency of allocation with finding new paths for the seeds chosen repeatedly in the exploitation stage

Evaluation

- Evaluate the efficiency of SPEM and AAPS
- Configuration:
  - choosing each best performance of **EcoFuzz, FidgetyAFL, FairFuzz,** and **AFLFast.new** on fuzzing **nm**
  - recording the energy allocated in random strategies of each turns, denoted as E_i, which i is the order of turn $(1 \leq i \leq N)$
  - recording the consumed energy for discovering the newest path of each turns, denoted as e_i, $0 \leq e_i \leq E_i$
  - recording the frequency of allocation with finding new paths for the seeds chosen repeatedly in the exploitation stage

Evaluation

- Evaluate the efficiency of SPEM and AAPS
- Configuration:
    - choosing each best performance of **EcoFuzz, FidgetyAFL, FairFuzz,** and **AFLFast.new** on fuzzing **nm**
    - recording the energy allocated in random strategies of each turns, denoted as E_i, which i is the order of turn $(1 \leq i \leq N)$
    - recording the consumed energy for discovering the newest path of each turns, denoted as e_i, $0 \leq e_i \leq E_i$
    - recording the frequency of allocation with finding new paths for the seeds chosen repeatedly in the exploitation stage

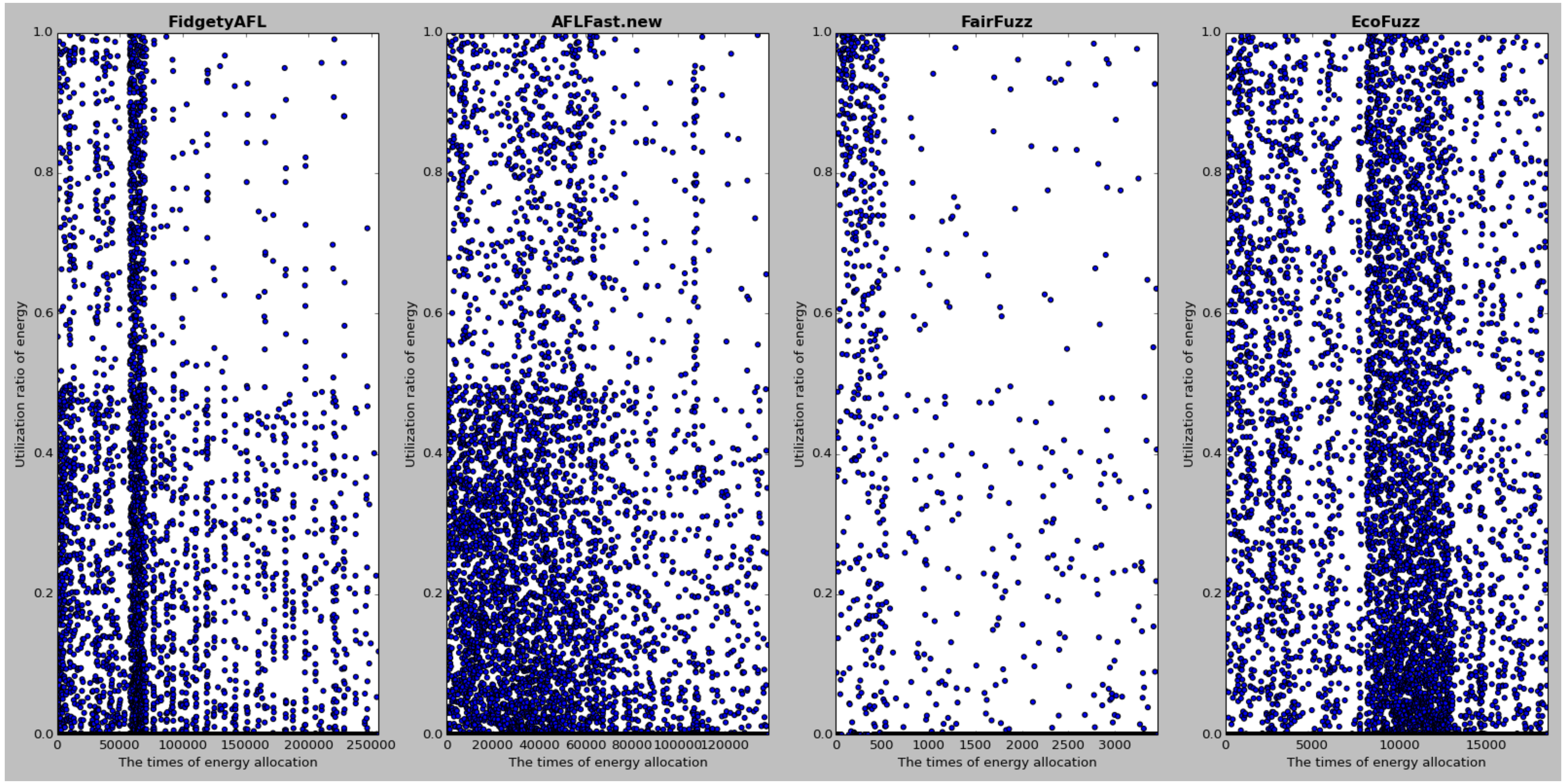
Evaluation

- Evaluate the efficiency of SPEM and AAPS
- Configuration:
  - choosing each best performance of **EcoFuzz, FidgetyAFL, FairFuzz,** and **AFLFast.new** on fuzzing **nm**
  - recording the energy allocated in random strategies of each turns, denoted as E_i, which i is the order of turn $(1 \leq i \leq N)$
  - recording the consumed energy for discovering the newest path of each turns, denoted as e_i, $0 \leq e_i \leq E_i$
  - recording the frequency of allocation with finding new paths for the seeds chosen repeatedly in the exploitation state

Evaluation

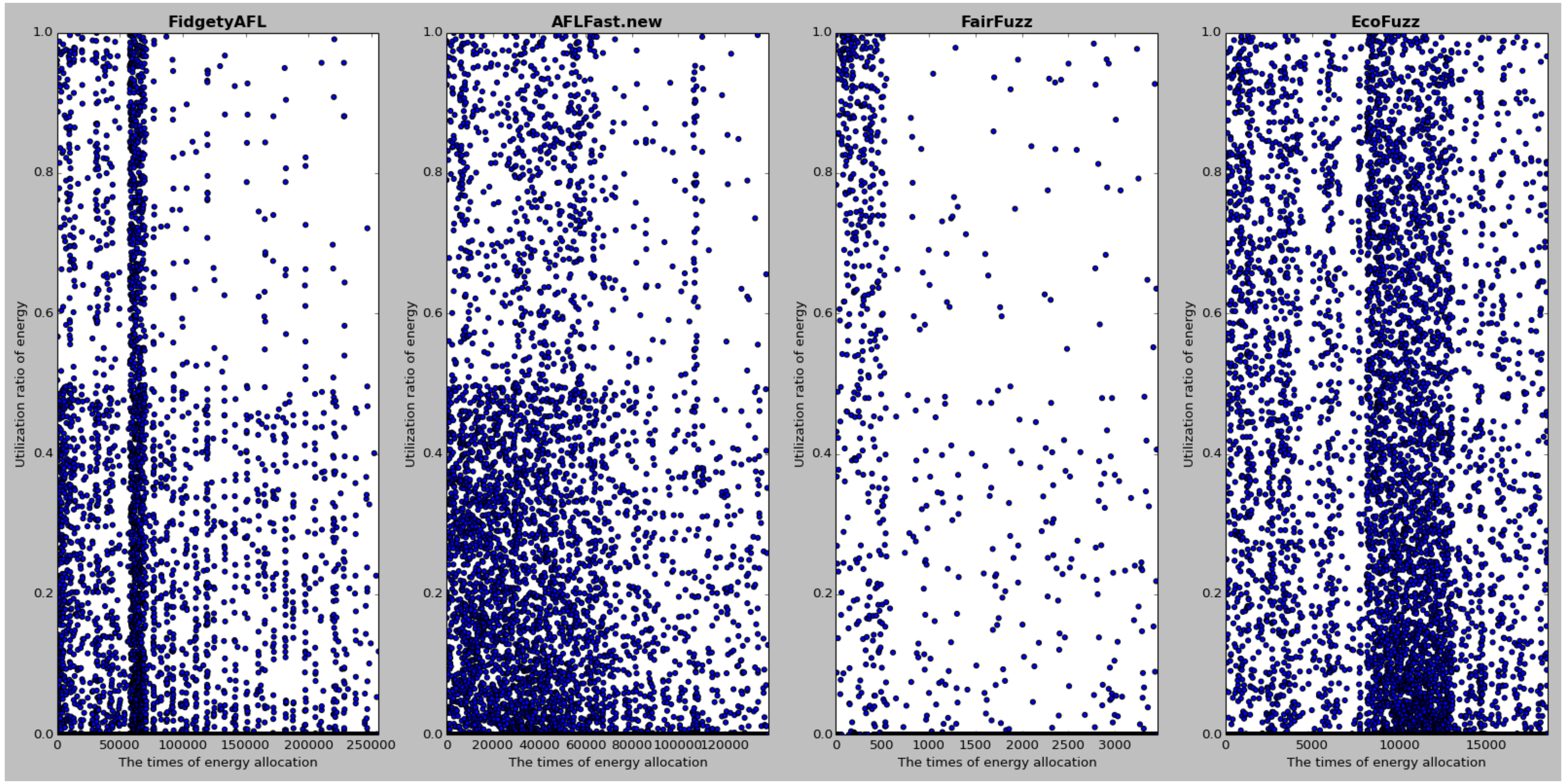
- Evaluation metric:
  - The **utilization ratio of energy** → $r_i = \frac{e_i}{E_i}$
  - The **average utilization ratio** → $\bar{r} = \frac{\sum_{i=1}^{i=N} r_i}{N}$
  - The **frequency of effective allocation** → $p = \frac{|\{i \mid e_i > 0, 1 \leq i \leq N\}|}{N}$

Evaluation


FidgetyAFL
AFLFast.new
FairFuzz
EcoFuzz
Utilization ratio of energy
The times of energy allocation
0.0
0.2
0.4
0.6
0.8
1.0
0
50000
100000
150000
200000
250000
20000
40000
60000
80000
100000
120000
500
1000
1500
2000
2500
3000
5000
10000
15000

Evaluation

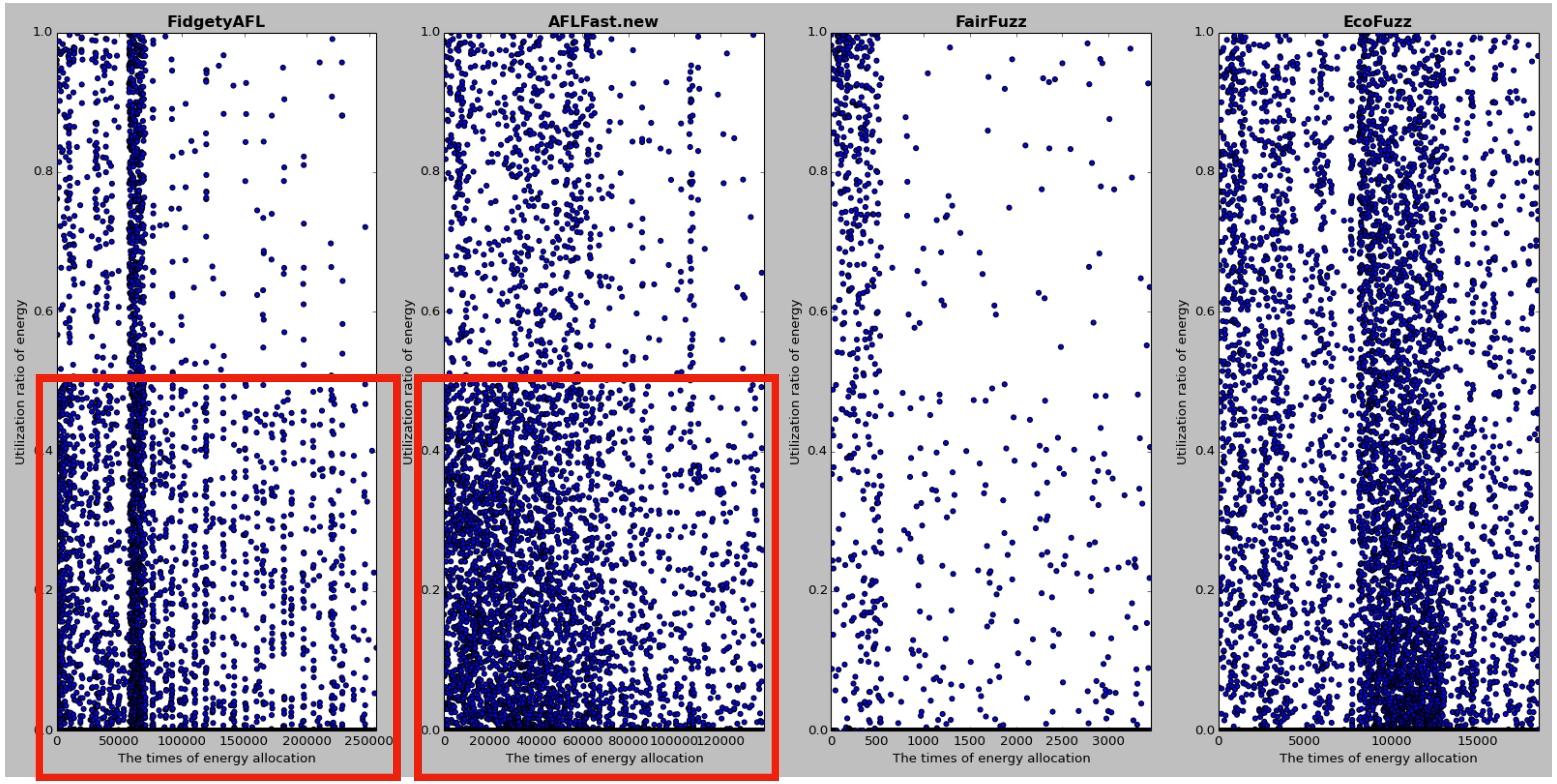
Scatter map of r_i with i



Evaluation

Scatter map of r_i with i

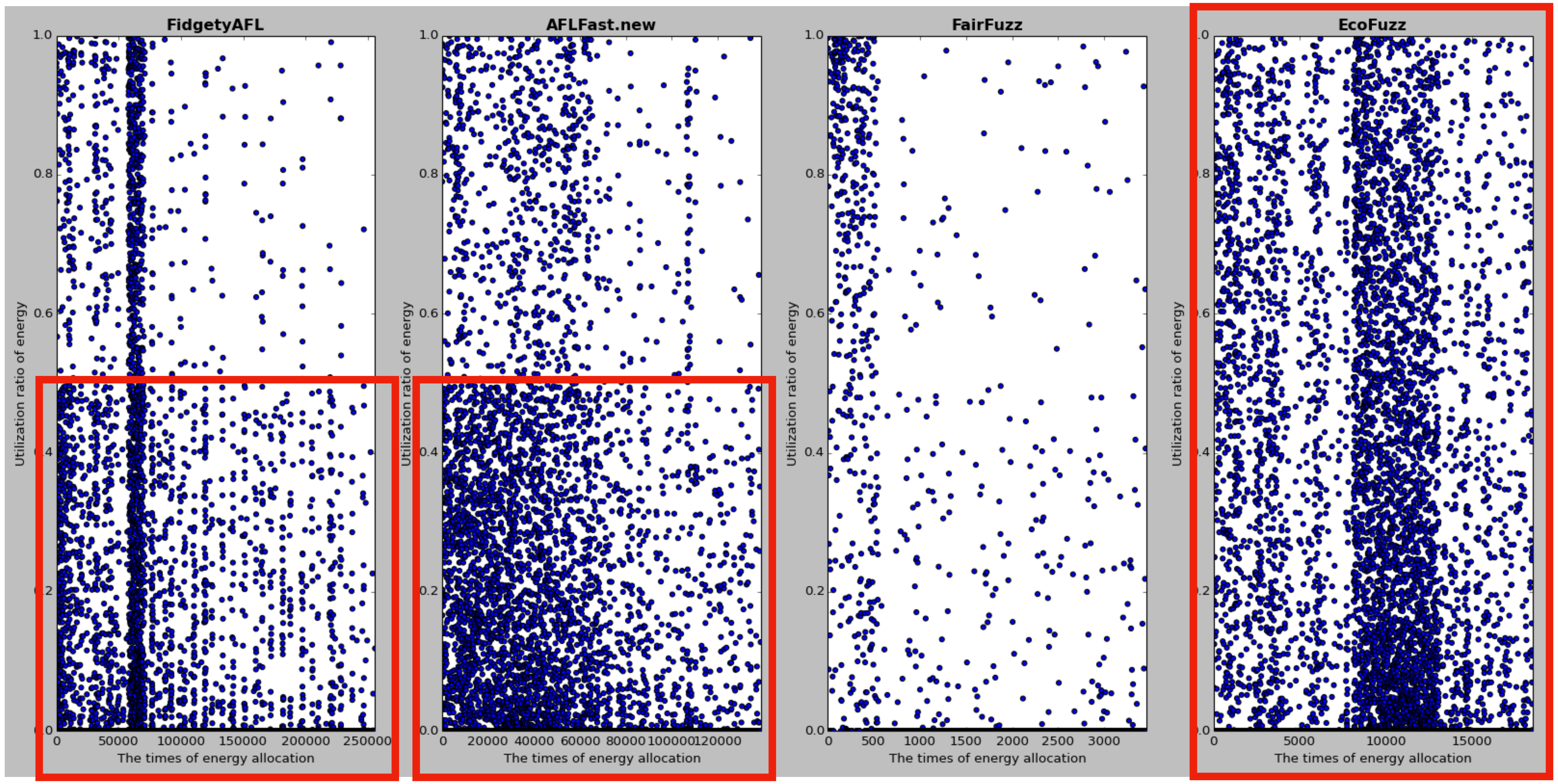
- **FidgetyAFL** and **AFLFast.new**
  - $r_i < 0.5$



Evaluation

Scatter map of r_i with i

- **FidgetyAFL** and **AFLFast.new**
  - $r_i < 0.5$
- **EcoFuzz**
  - $r_i \to 1.0$

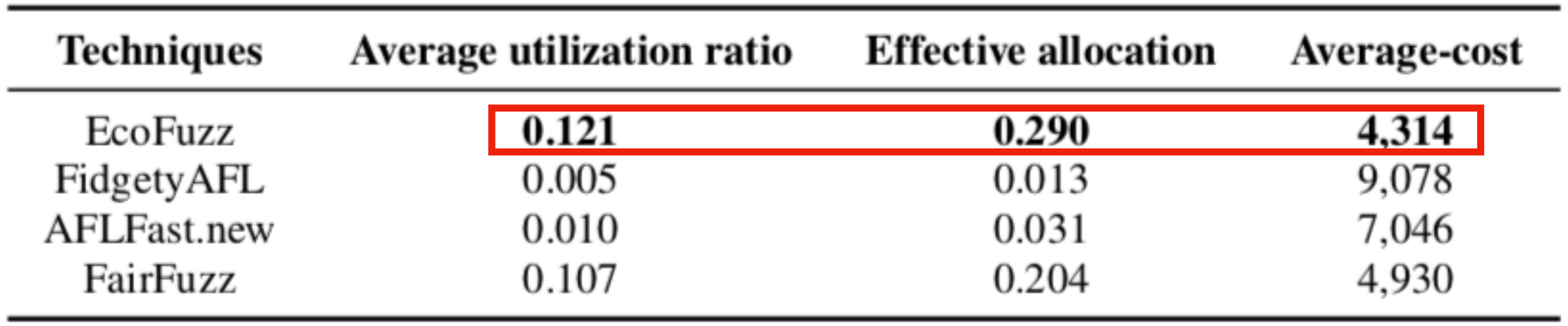


Evaluation

Table of $\bar{r}$ and p

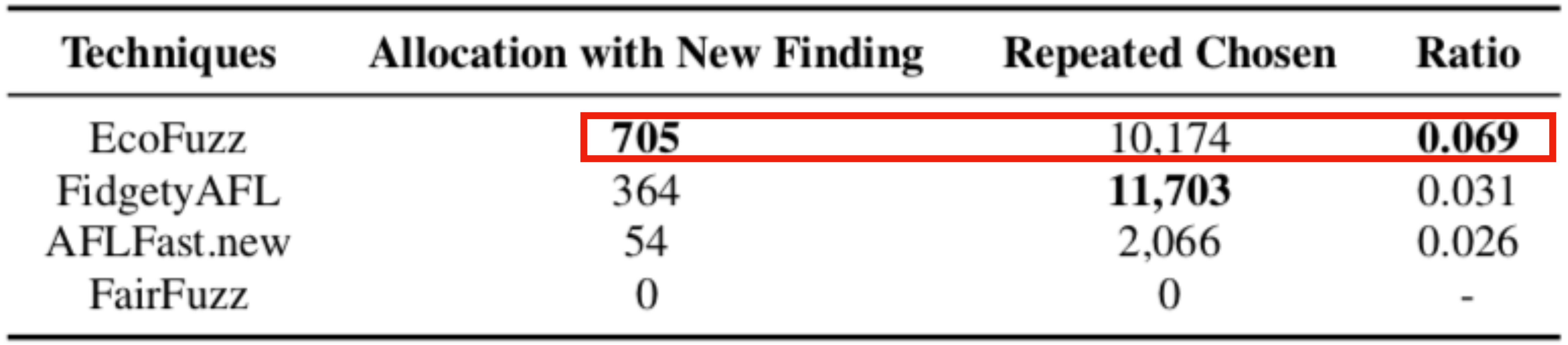
- EcoFuzz demonstrates the best performance
  - The **least average-cost**
  - The **highest average utilization**
  - The **highest frequency of effective allocation**
  - The **highest ratio** of effective allocation to the repeated chosen times in **exploitation state**

Table 3: The evaluation of power schedule



Techniques	Average utilization ratio	Effective allocation	Average-cost
EcoFuzz	**0.121**	**0.290**	**4,314**
FidgetyAFL	0.005	0.013	9,078
AFLFast.new	0.010	0.031	7,046
FairFuzz	0.107	0.204	4,930

Table 4: The evaluation of search strategy

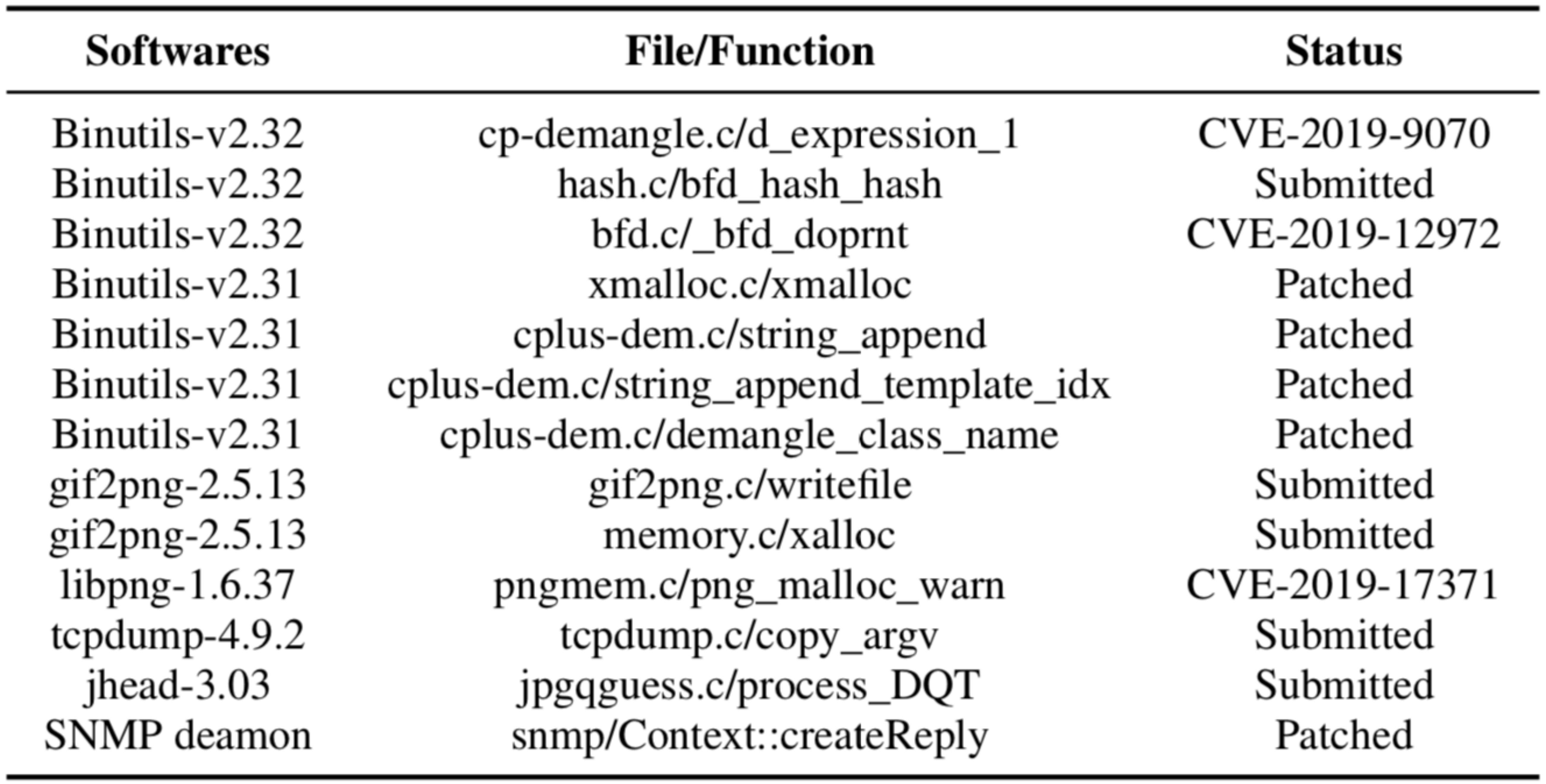


Techniques	Allocation with New Finding	Repeated Chosen	Ratio
EcoFuzz	**705**	10,174	**0.069**
FidgetyAFL	364	**11,703**	0.031
AFLFast.new	54	2,066	0.026
FairFuzz	0	0	-

Evaluation

- Detecting vulnerabilities
  - **12 vulnerabilities**
  - 2 CVEs

Table 8: The discovered vulnerabilities



Softwares	File/Function	Status
Binutils-v2.32	cp-demangle.c/d_expression_1	CVE-2019-9070
Binutils-v2.32	hash.c/bfd_hash_hash	Submitted
Binutils-v2.32	bfd.c/_bfd_doprnt	CVE-2019-12972
Binutils-v2.31	xmalloc.c/xmalloc	Patched
Binutils-v2.31	cplus-dem.c/string_append	Patched
Binutils-v2.31	cplus-dem.c/string_append_template_idx	Patched
Binutils-v2.31	cplus-dem.c/demangle_class_name	Patched
gif2png-2.5.13	gif2png.c/writefile	Submitted
gif2png-2.5.13	memory.c/xalloc	Submitted
libpng-1.6.37	pngmem.c/png_malloc_warn	CVE-2019-17371
tcpdump-4.9.2	tcpdump.c/copy_argv	Submitted
jhead-3.03	jpgqguess.c/process_DQT	Submitted
SNMP deamon	snmp/Context::createReply	Patched

Evaluation

- Detecting vulnerabilities
    - 12 vulnerabilities
    - **2 CVEs**

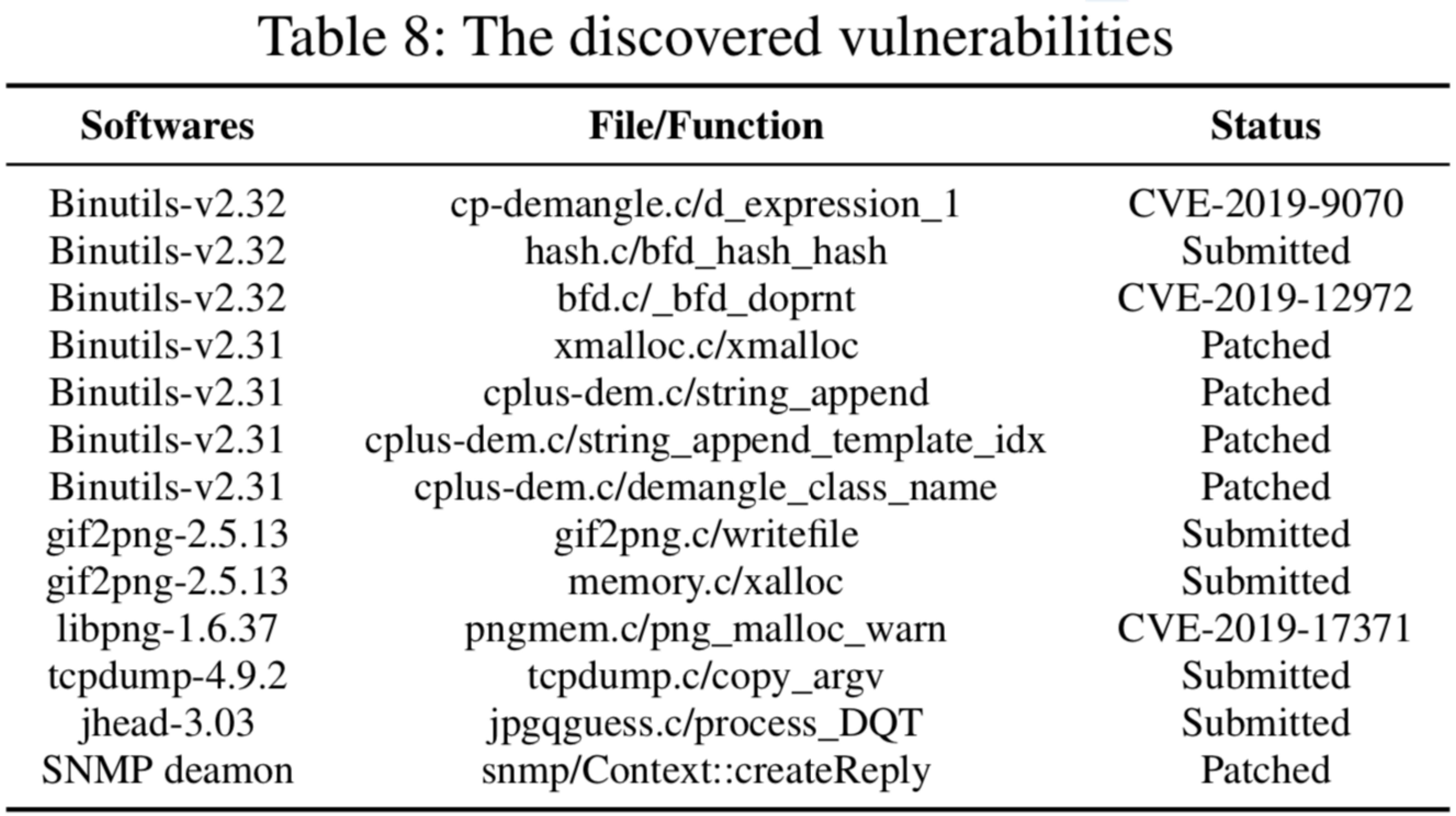


Table 8: The discovered vulnerabilities

Softwares	File/Function	Status
Binutils-v2.32	cp-demangle.c/d_expression_1	CVE-2019-9070
Binutils-v2.32	hash.c/bfd_hash_hash	Submitted
Binutils-v2.32	bfd.c/_bfd_doprnt	CVE-2019-12972
Binutils-v2.31	xmalloc.c/xmalloc	Patched
Binutils-v2.31	cplus-dem.c/string_append	Patched
Binutils-v2.31	cplus-dem.c/string_append_template_idx	Patched
Binutils-v2.31	cplus-dem.c/demangle_class_name	Patched
gif2png-2.5.13	gif2png.c/writefile	Submitted
gif2png-2.5.13	memory.c/xalloc	Submitted
libpng-1.6.37	pngmem.c/png_malloc_warn	CVE-2019-17371
tcpdump-4.9.2	tcpdump.c/copy_argv	Submitted
jhead-3.03	jpgqguess.c/process_DQT	Submitted
SNMP deamon	snmp/Context::createReply	Patched

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
    - Improving the reward probability of each seed:
      - FairFuzz(avoids mutating the crucial parts of seeds)
    - Establishing the mechanism of evaluating the seeds:
      - AFLGo(utilizes distance to evaluate each seed)
    - Optimizing the power schedule：
      - AFLFast(increases the assigned energy monotonically)

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
    - Improving the reward probability of each seed:
      - FairFuzz(avoids mutating the crucial parts of seeds)
    - Establishing the mechanism of evaluating the seeds:
      - AFLGo(utilizes distance to evaluate each seed)
    - Optimizing the power schedule：
      - AFLFast(increases the assigned energy monotonically)

Conclusion

- **VAMAB**
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
    - Improving the reward probability of each seed:
      - FairFuzz(avoids mutating the crucial parts of seeds)
    - Establishing the mechanism of evaluating the seeds:
      - AFLGo(utilizes distance to evaluate each seed)
    - Optimizing the power schedule：
      - AFLFast(increases the assigned energy monotonically)

Conclusion

- VAMAB
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
- **EcoFuzz**
  - Search strategy (Self-transition-based Probability Estimation Method)
  - Power schedule (Adaptive Average-cost-based Power Schedule)
- Comprehensive evaluation:
  - Attain 214% of the path coverage of AFL with reducing 32% test cases
  - Identifying 12 vulnerabilities

Conclusion

- VAMAB
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
- **EcoFuzz**
  - Search strategy (Self-transition-based Probability Estimation Method)
  - Power schedule (Adaptive Average-cost-based Power Schedule)
- Comprehensive evaluation:
  - Attain 214% of the path coverage of AFL with reducing 32% test cases
  - Identifying 12 vulnerabilities

Conclusion

- VAMAB
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
- **EcoFuzz**
  - Search strategy (Self-transition-based Probability Estimation Method)
  - Power schedule (Adaptive Average-cost-based Power Schedule)
- Comprehensive evaluation:
  - Attain 214% of the path coverage of AFL with reducing 32% test cases
  - Identifying 12 vulnerabilities

Conclusion

- VAMAB:
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
- EcoFuzz:
  - Search strategy (Self-transition-based Probability Estimation Method)
  - Power schedule (Adaptive Average-cost-based Power Schedule)
- **Comprehensive evaluation:**
  - Attain 214% of the path coverage of AFL with reducing 32% test cases
  - Identifying 12 vulnerabilities

Conclusion

- VAMAB:
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
- EcoFuzz:
  - Search strategy (Self-transition-based Probability Estimation Method)
  - Power schedule (Adaptive Average-cost-based Power Schedule)
- **Comprehensive evaluation:**
  - Attain **214%** of the path coverage of AFL with reducing **32%** test case
  - Identifying 12 vulnerabilities

Conclusion

- VAMAB:
  - Model the schedule process of CGF
  - Explain the exploration and exploitation in CGF
  - Point out how to improve greybox fuzzing
- EcoFuzz:
  - Search strategy (Self-transition-based Probability Estimation Method)
  - Power schedule (Adaptive Average-cost-based Power Schedule)
- **Comprehensive evaluation:**
  - Attain 214% of the path coverage of AFL with reducing 32% test case
  - Identifying **12** vulnerabilities

Thank you!

If you have some questions about our work, welcome to contact us!

Email: yuetai17@nudt.edu.cn

EcoFuzz: https://github.com/MoonLight-SteinsGate/EcoFuzz

National University of Defense Technology